Shared Control for Navigation and Balance of a Dynamically Stable Robot

By

Law Kwok Ho Cedric

A Thesis

Submitted in Partial Fulfillment of the Requirements

of the Degree of Master of Philosophy in

Mechanical and Automation Engineering

August 2001

Abstract

The single wheel, gyroscopically stabilized robot, called Gyrover, is a novel concept of mobile robots which provides dynamic stability for rapid locomotion. It has several advantages over statically stable multi-wheel robots including reduced sensitivity to attitude disturbances, complete recoverability from falling, and high dynamic stability. Further, this kind of robot can find obstacle-free paths on the ground more easily, and its narrow profile improves maneuverability. However, problems in steering and low-speed stability have kept such robot from becoming commonplace. In this thesis, the goal is to develop a semi-autonomous control for this kind of robots.

In order to provide a good foundation for the development of the robot, we partition the control problem into a set of loosely coupled computing modules (behaviors) by a behavior-based approach. Under this approach, we sort out two behaviors which locate at the lowest layer within the control architecture: (i) Lateral balancing, and (ii) Tiltup motion. These behaviors deal with the local instability problem in controlling a dynamically stable but statically unstable robot.

Since the robot concept brings a number of challenging issues in modeling and control by using some traditional control methods, therefore, we prefer to model the behaviors by learning. We propose using an efficient neural-network learning architecture that combines flexible cascade neural networks with extended Kalman filtering to capture the control skills from an expert operator. The models obtained

by the learning algorithm are validated by a stochastic similarity measure that is based on Hidden Markov Model analysis, which can compare the similarity between two dynamic and stochastic control trajectories.

Finally, we develop a shared control framework for the robot. Under the shared control, the control tasks are shared between the human operator and the automatic control system: the robot maintains local balancing, while the operator is responsible for the global navigation task. In order to let the system chooses between the control command from the two entities in an effective and systematic manner, a function is developed to tackle this problem.

Implementation results for the learning control and shared control are given in the thesis. The experiments demonstrate that this semi-autonomous approach provides a better way to control a dynamically stable but statically unstable robot, which can free the operator from being troubled by the low speed instability problem, and let him/her focuses on higher-level navigation tasks.

摘要

單輪陀螺平衡機器人，我們稱之爲 “Gyrover”，是一種新型的移動機器人。它可以實現機器人在快速運動下的動態平衡。此機器人比多輪式靜態平衡移動機具備更多優點，包括抗姿態干擾性，跌倒恢復能力，以及良好的動態穩定性。並且此種機器人能更容易找出無障礙物之行走路徑，它扁平的外型令其靈活性大大的提高。然而，其轉向以及低速行走的不穩定問題則令此類機器人不能受到廣泛的應用。因此，本篇論文的目的就是要爲此類機器人開發一個半自主的控制方法。

爲了提供一個良好的基礎予此系統的發展，我們透過一種基於行爲的控制方法，將整個控制問題分爲若干的非耦合計算模塊 (行爲)。藉此方法，我們得出兩個分佈在整個控制結構中最基層的行爲：(一) 側向平衡，及 (二) 跌倒恢復動作。這些行爲專門處理在控制一個動態平衡但靜態不平衡的機器人時所帶來的局部不穩定問題。

由於在傳統的控制方法下，這機器人帶來了一定的建模和控制上的挑戰性問題，所以我們嘗試利用機器學習的方法來建模。我們提出使用一種有效的神

經網絡學習結構，此結構結合了級聯人工神經網絡及結點解耦延伸卡爾曼過濾法來吸取專業控制人員的控制技巧。我們並且使用基於隱含馬爾卡夫模型的隨機相似性量度指數來驗證學習模型的可靠性，此量度指數能夠比較動態及複雜的軌跡的相似度。

最後，我們爲此機器人開發一個共享控制架構。在此控制模式中，不同的控制工作將被分配予控制人員以及自主控制系統：自主控制系統負責維持機器人的局部平衡狀態，而控制人員則負責全局導航的工作。爲了令整個系統能有效地選擇執行控制人員或自動系統的命令，我們建立了一個函數來應付此問題。

此論文中提供了學習控制和分配控制的驗証結果。實驗結果証明這半自主的控制模式能提供一個更好的方法來控制一個動態平衡但靜態不平衡的機器人，令控制員大大減少對其在低速行走時之穩定性的關注，從而令他/她更能專注於高層次的導航工作。

Acknowledgments

I sincerely thank my supervisor, Prof. Yangsheng Xu, for his friendship and intellectual guidance on all aspects of the work. He has given me a lot of motivations and opportunities to learn throughout the two years of study.

I would like to thank H. Ben Brown and his team at Carnegie Mellon University for providing excellent technical support for Gyrover. Special thanks go to Arne J. Suppe, who has spent plenty of time to solve the software problem in the robot. Also thanks go to Samuel K. W. Au for his preliminary work in the wireless communication and the basis of the programs in Gyrover, which provides a good foundation for me to work on.

In addition, I thank the following people for their friendship, both personal and professional, throughout my two years study: Allan Lam, Mark Lau, H. C. Lo, Yong He and Eric Wong. They have given me a lot of supports during the toughest period, and they have spent many nights to work with me, which are memoriable for me. Also, thanks go to my colleagues in the Advanced Robotics Lab. in CUHK, especially K. K. Lee and Ou Yongsheng.

Finally, with great love and respect, I acknowledge the support, love and encouragement of my family, without whom, I surely would not have reached this stage in my life. Special thanks also go to Welly Lau for her support and concern.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Land locomotion can be broadly characterized as quasi-static or dynamic. Quasi-static equilibrium implies that inertial effects are small enough to ignore. The motions are slow enough that static equilibrium can be assumed. Traditionally, mobile robots are treated as quasi-static devices. Numerous robots with multiple wheels or legs have been developed to maximize their mobility on various terrain. Generally, these robots have featured low center of mass and broad bases of support, along with intelligent control algorithms designed to keep the center of mass gravity vector within the support ploygon. Although these robots are statically stable, they are often limited by motion-planning constraints and hence are usually designed for relatively low-speed operation. Dynamic factors have little influence on such systems and consequently have been ignored.

On the other hand, consider a bicycle or a motorcycle which has two wheels in the fore-aft configuration. Such vehicle is statically unstable in the roll direction, but can achieve dynamic stability at moderate speed through an appropriate steering geometry and gyroscopic action of the steering front wheel. Steering stability increases

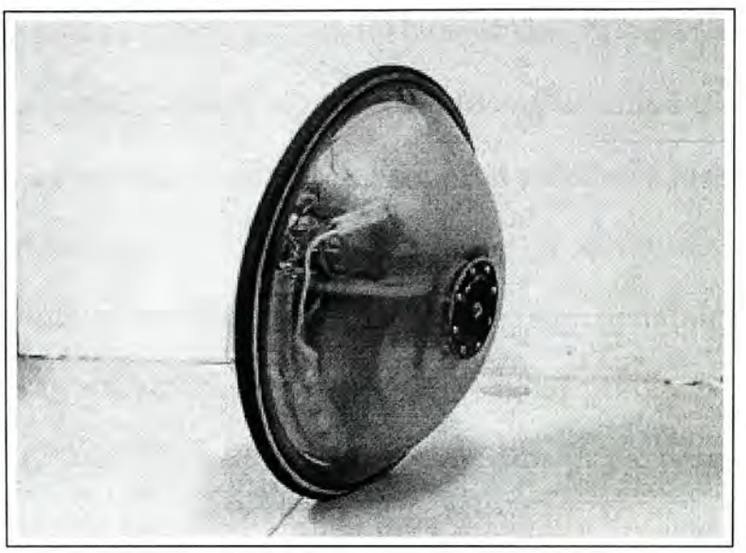


Figure 1.1: The Gyrover.

with speed gradually due to gyroscopic effects. Dynamic forces at the wheel-ground contact point act on or near the vehicle center (sagital) plane, and thus produce minimal roll disturbances. Additional, bicycles have greater maneuverability than the quasi-static devices.

As a logical extension of this argument, in order to retain static (quasi-static vehicles) and dynamic (bicycles) stabilities, we designed a single wheel, gyroscopically stabilized robot, Gyrover, as shown in Figure 1.1.

Gyrover is a single, large-diameter wheel that relies on gyroscopic action for dynamic stability. In its simiplest form, Gyrover is a large wheel with its propulsion, steering and other equipment suspended from its axle. The rotational motion of the wheel gyroscopically stabilizes its attitude, while directional control is accomplished by reacting against an internal gyroscope, to produce "lean steering". An internal gyroscope may also augment the lateral stability of the robot, and allows it to stand and turn in place.

Owing to the gyroscopic effect of the spinning flywheel, the static stability of the

robot is greatly improved. Dynamic disturbances due to surface irregularities act through or near the wheel's center of mass, producing minimal torques in roll, pitch and yaw. In terms of attitude control, the robot is relatively insensitive to fore/aft and side slopes. Although Gyrover has a number of advantages over traditional wheel robots, its complex dynamics characteristics (e.g. dynamic coupling between the wheel and the flywheel) bring certain challenging problems in modelling and control at the same time. We will further explain this in next part.

Thus far, Gyrover is being controlled only manually, by using two joysticks to control the drive and tilt motors through a radio link. Even for human operators, the contol task of Gyrover is very difficult due to its inherent instability in its lateral (roll) direction. Consider a human riding a unicycle, the rider needs to concern the lateral stability of the vehicle. To keep steering the vehicle is also a problem since it does not have any proper steering mechanism visually, from the concept of gyroscopic precession, the rider needs to lean on one side to achieve steering. It would be difficult if the speed of the system is too slow for it to gain enough dynamical stability.

In this thesis, our goal is to develop a semi-autonomous control system for Gyrover. As an extension of our work in [14], we model the human control skills in balancing and tiltup the robot in the vertical position using a machine learning algorithm. By the success in implementing the learnt models, we develop a shared control framework for the robot in this thesis. This work is definitely unique and original from other related researches.

First of all, we propose using a behavior-based control approach to breakdown the control problem of Gyrover. For each of the low level task (e.g. lateral balancing), although it is difficult to develop an accurate dynamic model, we observed

that human beings are excellent in mastering complex system such as car driving. Therefore, we use a model-free machine learning algorithm, Cascade Neural Network (CNN) with Node-Decoupled Extended Kalman Filtering (NDEKF), as a modelling tool for human control skill learning in Gyrover. Due to the limitation of the number of the sensors on board, it is impossible for us to implement navigation control at this moment. This limitation motivates us to develop a shared control framework for Gyrover, that is, while the operate is giving a navigation command, the robot will remain the lateral stability along the journey. This reduce the operator's effort in controlling the robot significantly.

1.2 Related work

The modelling of this highly coupled, dynamically stable system is a very challenging problem. Several researhers have been attemped to develop a dynamic model for the control of Gyrover. As a first step in modelling this complex system, a 3-dimensional model of the wheel part of the Gyrover was developed and discussed in [2], utilizing the constrained Lagrangian principle for nonholonomic system. Implementations of the equations of motion in a real-time graphic simulator and the simulation of the dynamic behavior of the wheel for different initial conditions and different gravitational effects were also presented in [2].

However, due to the motion between the flywheel and the robot is highly coupled with each other, it is necessary to consider the dynamics of the single wheel and the spinning flywheel at the same time. By taking the actuation of the flywheel inside the robot into account, the dynamic behavior and the nonholonomic constraints of the systems were also investigated [4].

The dynamic model developed in [2, 4] is further simiplify by decoupling the

model with respect to the control input. Based on the linearization, the motion control is decomposed into three parts: (1) controlling the rolling speed, (2) controlling the tilting variable, and (3) a linear state feedback controller to control the lean angle of the robot, so as to track a circular path or a straight line [5]. Further, a line following controller for tracking any desired straight path is developed in [6]. The controller is divided into two parts : (1) velocity control and (2) torque control.

Another version of the dynamic model of Gyrover is developed based on the Newton-Euler approach. The linearized model is used to develop a state feedback controller. The design methodology is based on a semi-definite programming procedure which optimize the stability region subject to a set of Linear Matrix Inequalities that capture stability and pole placement constraints. Finally, the controller is combined with the extended Kalman filter. [3].

Moreover, the dynamics and control for the robot to roll on an inclined plane is studied in [8, 9]. The effect of internal pendulum motion and the inclination angle of the plane are also addressed. The condition of rolling up an inclined plane is figured out and different motion strategies are proposed when it has violated the rolling up condition.

Finally, a complete different control approach is used in [14]. This was a preliminary work in abstracting the human strategy in controlling a dynamically stabilized robot.

1.3 Thesis overview

As mentioned in the previous section, the control of Gyrover is heavily relied on the dynamic model of the robot. However, due to the complexity of the system (highly coupled dynamics and nonholonomic nature), the proposed dynamic model is much

being simplified and thus incomplete. Many of our researches are still focusing on the dynamics and control of the robot. At the meantime, we are seeking for other modelling method which enables us to develop a control method for the robot. We found that machine learning is an alternative for us to achieve the goal since no a priori model is required for the learning process.

Therefore, this thesis applies machine learning techniques towards abstracting and implementing the models of human control strategy in real Gyrover control. However, due the limited of sensors available on the robot, it is impossible for us to develop a fully autonomous system at this stage. To this end, with the idea of shared control, a degree of control can be shared to the machine. Therefore, in a shared control environment, the human operator will entirely responsible for the navigation control on the robot, while the machine will responsible for some local stability tasks.

This thesis is organized as follows:

- Chapter 2: **Single wheel robot: Gyrover**

  A detail description about the Single Wheel Robot will be given in this chapter. First of all, we introduce the history of the development of the robot. Next, the hardware components and the robot's concept are discussed. Later, we study the effects of the internal flywheel. Since the flywheel is a very important component in Gyrover, with a better understanding of the flywheel, a better control of the robot would result. Finally, we summarize some characteristics in Gyrover control which are different from the traditional mobile robots.

- Chapter 3: **Learning control**

  Since Gyrover is a complex system in both dynamics and control, we have

many difficulties in deriving an accurate model for the robot by using traditional control method. Therefore, a model-free machine learning algorithm, an alternative control method, will be discussed in this chapter. Moreover, we propose using a similarity measure to validate the learnt models.

- Chapter 4: **Control architecture**

  In this chapter, we propose using a subsumption architecture for controlling Gyrover in a complex environment. The subsumption architecture is a special case of behavior-based control for robotics. Behavioral modules are added as "layers" with each layer performing a complete behavior. We first decompose the control problem in Gyrover into many behavioral modules, to develop the subsumption architecture, low level behavioral modules are arranged at the bottom and those in higher level is built on top of lower levels. By using this approach, we are able to have a clear picture for the autonomous control problem in Gyrover. Later, we will discuss the behaviors we are going to model within the overall control structure. A detail discussion will be given in the last section of this chapter.

- Chapter 5: **Implementation of learning control**

  The casade neural network models for the motions of lateral balancing and tiltup are implemented in this chapter. The models are validated by a similarity measure first, by comparing the trajectories generated from the models and those from human operator. Next, implementation results of the individual models will be given. From the experimental results, we observe that each model is subjected to some initial condition. For instance, the tiltup model is unable to balance the robot into the vertical position after tiltup from the

ground. To address this problem, we combine the two motions into a single motion. Experiments show that the combined motion can fully recover the robot from the fall position and stabilized at the vertical position for a long period of time.

- Chapter 6: **Shared control**

  Due to the complexity of the Gyrover system, we are unable to develop a fully autonomous system for the robot yet. Although a large portion of control is still rely on online human operator, in order to reduce the workload of the operator, we propose a shared control framework for Gyrover. To effectively and accurately distribute the workload in the control, a decision function is developed in a shared control system in this chapter. A number of experiments are conducted to verify the algorithm.

- Chapter 7: **Conclusion**

  A summary of contributions of the thesis is given in this chapter. A number of suggestions for the future development of Gyrover are also included.

Chapter 2

Single wheel robot: Gyrover

2.1 Background

Gyrover is a novel, single wheel gyroscopically stabilized robot, originally developed at Carnegie Mellon University, in August, 1992. 3 prototypes have already been developed. Figure 2.1 and Figure 2.2 shows the first and the second prototypes respectively. The latest model of Gyrover (the third generation) is shown in Figure 2.3.

In the literature, there are precedents for single-wheel-like vehicles. In 1869, R.C. Hemmings patented "Velocipede", a large wheel encircling the rider, powered by hand cranks. Palmer describes several single-wheel vehicles with an operator riding inside. A 1935 publication describes *Gyroauto*, which carried the riders between a pair of large, side-by-side wheels, and was claimed capable of a speed of 116 mph. In, a concept having a bus-like chassis straddling a huge central wheel was also described.

Before the first prototype of Gyrover was developed, several alternative configurations had been considered, such as, a spherical shape, two wheels side by side and outboard wheels configuration. However, most of the above designs do not exhibit

natural steering behavior resulting from the interaction of the gravitational torque and the gyroscopic effect. The present concept, with all the sensors and instruments enclosed in the singlw wheel, provides a simple, reliable and rugged vehicle.

Figure 2.1: Gyrover I.

Figure 2.2: Gyrover II.

Gyrover I has a diameter of 29 cm and a mass of 2.0 kg. It can be easily driven and steered by remote control, has a good high-speed stability on smooth or rough terrain, and can be kept standing in place. The main shortcomings of this robot are its lack of resilience and vulnerability to wheel damage, excessive battery drain due to drag on the gyro, inadequate torque in the tilt servo and incomplete enclosure of the wheel. Gyrover II was designed to address these problems. It is slightly larger than Gyrover I (34 cm diameter, 2.0 kg) and uses many RC model parts. Tilt-servo torque and travel were approximately doubled. The robot contains a variety of sensors to monitor motor current, position and speed, tire and vacuum pressure, body orientation and gyro temperature.

The latest version, Gyrover III, was designed on a larger scale to premit it to carry numerous inertial sensors and a computer (486PC) for data acquisition and

Figure 2.3: Gyrover III.

control. This machine uses a lightweight 40 cm bicycle tire and rim and a pair of transparent domes attached to the axle. The overall weight is about 7 kg. This prototype is readily for us to implement some control algorithms into the robot to develop a semi/fully autonomous control system. However, vision is still not avaiable in this prototype yet.

2.2 Robot concept

The actuation mechanism in Gyrover consists of three seperate actuators: (1) a spin motor, which spins a suspended flywheel at a high rate, imparting dynamic stability to the robot; (2) a tilt motor, which controls the orientation of the flywheel; and (3) a drive motor, which causes forward or backward acceleration, by driving the single wheel directly.

$$T = J\omega \times \Omega \tag{2.1}$$

where T is the applied torque normal to the spin and precession axis, J is the wheel polar moment of inertia about the spin axis, ω is the angular speed of the wheel,

and Ω is the wheel's precession rate normal to the spin axis.

The behavior of Gyrover is based on the principle of gyroscopic precession, equation(1.1), as exhibited in the stability of a rolling wheel. Because of its angular momentum, a spinning wheel tends to precess at right angles to an applied torque (classical gyroscopic pression). Thus, if a torque (T) is applied about the wheel's longitudinal axis, rather than falling over, the wheel precesses about the vertical axis, causing it to follow a curved path. If the wheel leans to one side, the gravitationally induced torque causes it to precess so that it turns in the direction it is leaning, tending to stabilize its upright position.

Gyrover supplements this basic concept with the addition of an internal gyroscope nominally aligned with the wheel and spinning in the direction of forward motion. The gyro's angular momentum produces lateral stability when the wheel is stopped or moving slowly. A tilt mechanism enables tilting the gyro's axis about the fore/aft axis with respect to the wheel. Because the gyro acts as an inertial reference in attitude, the immediate affect of the tilt action is to cause the wheel to lean left or right, which in turn causes the wheel to steer (precess) in the direction of leaning. Torques generated by a drive motor, reacting against the internal mechanism which hangs as a pendulum from the wheel's axle, produce thrust for accleration and braking.

Gyrover has a number of advantages over multi-wheeled vehicles:

1. The entire system can be enclosed within the wheel to provide mechanical and enviromental protection for the equipment and actuation mechanism.

2. Gyrover is resistant to getting stuck on obstacles because it has no body to hang up, no exposed appendages, and the entire exposed surface is driven.

3. The tiltable spinning flywheel can be used to right the vehicle from its statically stable, rest position (on its side). The wheel has no backside on which to get stuck.

4. Gyrover can turn in place by simply leaning and precessing in the desired desired direction with no special steering mechanism, which enhance maneuverability.

5. Single-point contact with the ground eliminates the need to accommodate uneven surfaces and simplifies control.

6. Full drive traction is available because all the weight is on the single drive wheel.

7. A large pneumatic tire may have very low ground-contact pressure, resulting in minimal disturbance to the surface and rolling resistance.

Although the robot offers tremendous potential applications, the robot concept also brings a number of challenging problems in modeling and control due to the following characteristics:

- Dynamic coupling: It is a highly coupled dynamic system between the wheel and the flywheel because the flywheel is mounted on the rolling wheel through a 2-link manipulator. In fact, there is no actuator to control the roll angle of the robot directly, the system only allows us to control its roll angle indirectly by tilting the orientation of the spinning flywheel.

- Nonholonomic constraints: The single wheel robot is subject to two nonholonomic constraints: the first order and the second order nonholonomic constraints. The first order constraint is based on the assumption that the robot

rolls on a plane without slipping. The second order one is due to underactuation in the roll direction.

- Unstable in lateral direction: Similar to a single track vehicle such as bicycle or unicycle, the robot is inherently unstable in the lateral direction.

- Gyroscopic stabilization: A characteristic of gyroscopic stabilization, not generally understood, is that the stability depends on the freedom to precess. For our case, a gyro with horizontal axis normally precesses about the vertical (yaw) axis when a torque is applied about the fore/aft (roll) axis. If the yaw precession is prevented by some obstruction, a yaw torque will be generated that completely negates the stabilizing effect, which makes the wheel to fall like a static, rigid body. If the precession is resisted by a yaw torque, the unpright attitude will decay as the wheel precesses, and it will fall slowly in the direction of the roll torque.

These are the reasons why we prefer using a model-free approach to control the robot rather than classical control method which requires ultimate understanding about the dynamic properties of the system.

2.3 System description

In this section, details of Gyrover's sensing, actuating mechanisms and computing device are discussed. The latest model we are using currently is Gyrover III. It is built with a light-weight bicycle tire and rim and a set of transparent domes. It includes a radio system for remote control, on-board computer and a number of sensors to permit data-logging and on-board control of the machine's motion.

There are 3 actuating mechanism in Gyrover: (i) Gyro tilt servo, (ii) Drive motor, and (iii) DC gyro motor. Table 2.1 gives a detail description of each of them.

Actuator	Symbol	Descriptions
Gyro tilt servo	u_0	The tilt servo controls the relative angle of the gyro spin axis with respect to the wheel axis. In fact, by controlling the tilt servo, we are able to controls the lean angle angle of the robot indirectly.
Drive motor	u_1	The robot forward/backward drive system uses a 2-stage, tooth belt system to amplify the torque from the drive motor.
DC gyro motor	u_2	This motor cause the internal gyro to spin at a dersirable operating speed, increase the angular momentum of the gyro.

Table 2.1: Table of different actuating mechanism in Gyrover.

A number of on-board sensors have been installed on Gyrover to provide information about the states of the machine to the on-board computer. The information includes:

- Gyro tilt angle, β_a
- The servo current
- Drive motor current
- Drive motor speed
- Gyro speed, $\dot{\gamma}_a$
- Angular rate (3-axes: Roll-Pitch-Yaw), $\dot{\beta}$, $\dot{\gamma}$ and $\dot{\alpha}$
- Accleration (3-axes: Roll-Pitch-Yaw), $\ddot{\beta}$, $\ddot{\gamma}$ and $\ddot{\alpha}$
- Robot tilt angle (Roll), β

All these signals, plus the control inputs from the radio transmitter, can be read by the computer. A custom-built circuit board contains the control computer and flashdisk, interface circuitry for the radio system and servos, components and logic to control power for the actuators, and an interface for the on-board sensors. The on-board processing is performed by a 486 Cardio PC.

In addition, several more sensors are planned to incorporate with our control algorithms in the near future. Visual processing capability or a Global Positioning System (GPS) is a big issue for the autonomous control, however, due to the structural limitation of the robot, we have not equipped the robot with this kind of device yet.

2.4 Flywheel characteristics

In this section, we are going to study how the orientations of the internal flywheel affect the Gyrover's motion. By having a better understanding of this problem, humans can control the robot more effectively. First of all, let's consider the case of a rolling disc, according to the fundamental equation for gyroscopic precession (2.1), the idea is illustrated in Figure 2.4.

For instance, given that the disc is rolling on a plane at its upright position, if a torque is applied to the X-axis, an angular rate of precession will be induced at the Y-axis. Therefore, rather than falling over, the disc will turn in the direction it is leaning, tending to stabilize its upright position.

Gyrover is considered as a combination of three components: (1) a wheel, (2) an internal mechanism, and (3) an internal flywheel. The robot is so designed that the intial orientation of the flywheel is located at 0^o ($\beta_a = 0$), with the spinning axis parallel to the pitch direction of the robot, Figure 2.5 and 2.6. Moreover, the

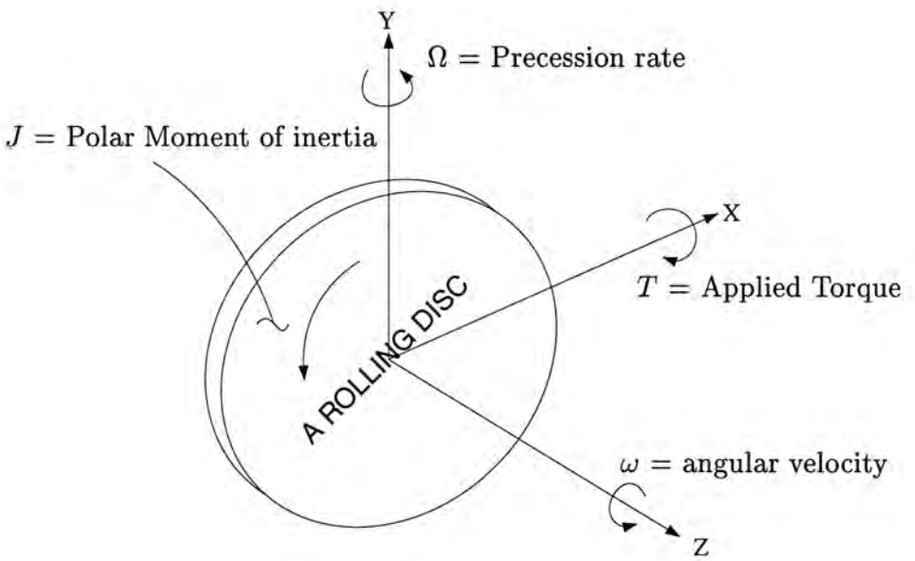


Figure 2.4: The fundamentals of gyroscopic precession.

orientation of the flywheel is bounded between $\pm 90^o$. At the boundary conditions ($\beta_a = \pm 90^o$), the spinning axis of the flywheel will be paralleled to the yaw direction of the robot.

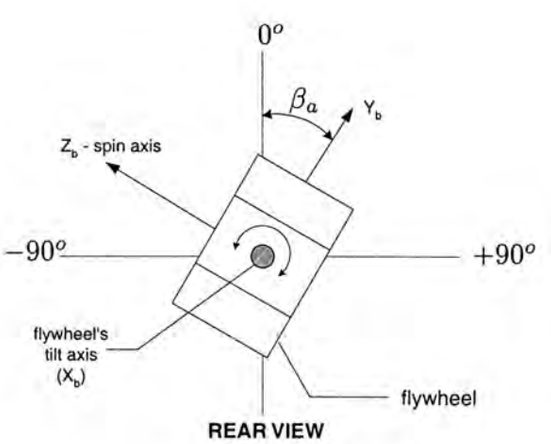


Figure 2.5: Flywheel's orientation is limited to $\pm 90^o$.

The high speed spinning flywheel, when installed in Gyrover, its angular momentum can provide lateral stability when the robot is moving slowly or even remain at a stationary location. Consider that the flywheel is located at $\beta_a = 0$, if we applied a torque along the tilt axis (X_b), from equation (3.1), a torque will be induced at

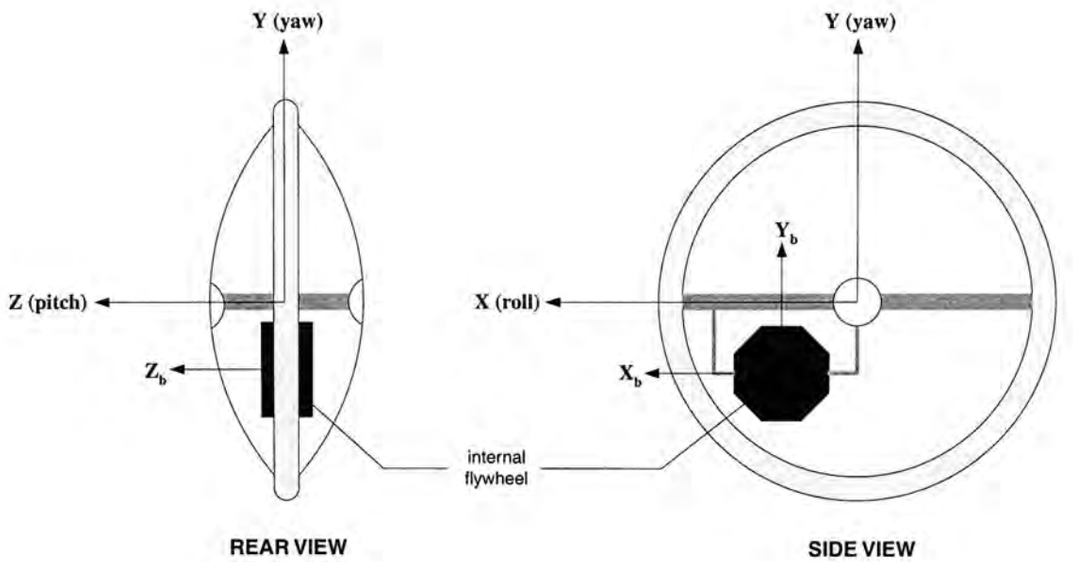


Figure 2.6: Coordinate frames of the Gyrover and the flywheel.

the direction Y_c. Since the flywheel is attached to the robot with the motion at Y_b is fixed, the torque results in the coupling motion between the yaw and roll axis of the robot. By this coupling motion, the gyroscopic torque from the flywheel can balance the gravitational torque which intend to make the robot fall down.

However, when the flywheel's spinning axis is in parallel (or closely parellel) with the wheel's yaw axis, the torque produced by the flywheel will no more contribute to stabilize the wheel in the lateral (roll) direction. On the other hand, the torque will contribute to the internal mechanism of the robot which will cause undesirable motion to the whole system. This can be demonstrated by the following experiments.

Besides the above problem, there is another disadvantage if the flywheel's spinning axis is in parallel with the robot's yaw axis, i.e. when $\beta_a = \pm 90^o$. Due to the hardware limitation, the tilt angle of the flywheel is bounded by:

$$-90^o \leq \beta_a \leq 90^o \tag{2.2}$$

Consider that if $\beta_a \approx 90^0$, since the flywheel cannot be tilted further beyond 90^o,

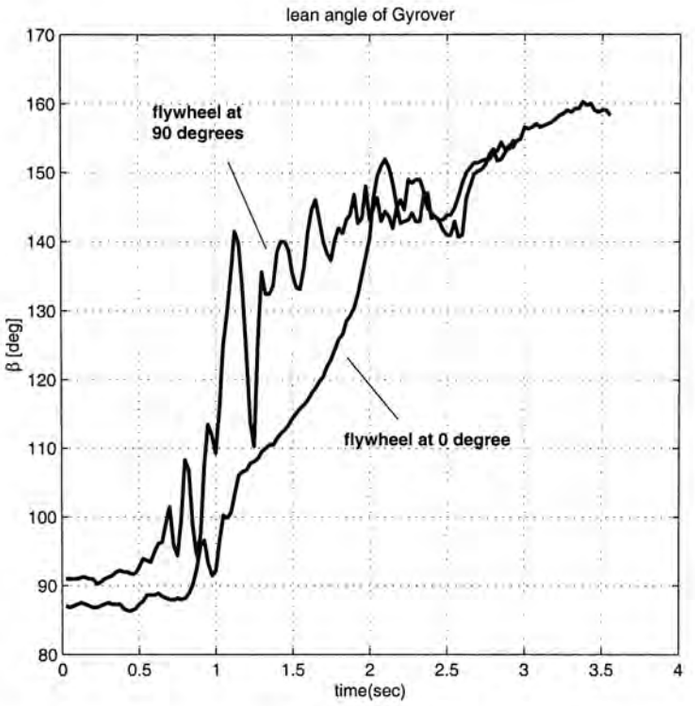


Figure 2.7: The effects of the flywheel in Gyrover.

if the robot keeps falling on a particular side, the flywheel is unable to generate a sufficient torque to oppose the change, the robot will fall down eventually. The case is similar when $\beta_a \approx -90^o$.

Thus, during the control of Gyrover, we should avoid the flywheel to stay at or near $\pm 90^o$ as possible as we can. In other words, in order to stabilize the robot and to response to any disturbance in the lateral direction effectively, we should always keep the flywheel to remain at 0^o. This can avoid the motion of the internal mechanism, which is undesirable, the flywheel is also able to provide maximum degree of freedom (DOF) to oppose any changes in the lateral direction of the robot. Here, we introduce a method to measure the DOF of the flywheel:

$$DOF_{flywheel} = 1 - \left| \frac{\bar{\beta}_a}{\beta_{a:max}} \right| \tag{2.3}$$

where $\bar{\beta}_a$ is the mean tilt angle of flywheel, $\beta_{a:max}$ is the maximum tilt angle the

flywheel can achieve (i.e. $\pm 90^o$) and $0 \geq DOF_{flywheel} \geq 1$.

If $DOF_{flywheel}$ is equal to 1, that means the flywheel is always located at the 0^o position. If $DOF_{flywheel}$ is equal to 0, implies that the flywheel is always located at $\pm 90^o$, which is not desired. Therefore, under this measurement, the greater value of $DOF_{flywheel}$, the better control of the robot (flywheel) would result. For the experiments we conduct later, we use this measurement to evaluate the "quality of control" of Gyrover.

2.5 Control patterns

Conventional mobile robots constitute the following behaviors during navigation: (i) Obstacles aviodance, (ii) Object recognition (image processing behavior), (iii) Path planning, (iv) Path tracking, and (v) Wondering (randomly move around). Besides the traditional mobile robot behaviors, Gyrover has some other behaviors which are different from them.

- **Lateral balancing**

  Lateral stability is the most basic problem of a single wheel vehicle, especially when the wheel does not roll, which is similar to a bicycle. The robot is inherently unstable in the lateral direction because there is no actuator which directly balance itself. However, since a spinning flywheel is mounted on the rolling wheel through a two-link manipulator, by tilting the internal flywheel into different orientation, we are able control the robot in the lateral direction indirectly.

- **Fall recovery**

Fall recovery is an unique ability of Gyrover when compared with other mobile robot. Although the robot is unstable in lateral direction, which implies that it may sometimes fall on the ground, it is able to recover from the fall positions by controlling the orientation of the flywheel. Gyrover resists to get stuck on obstacles because it has no body to hang up, no exposed appendages, and the entire exposed surface is live. The wheel also has no backside on which to get stuck.

- **Heading control**

  Since the robot do not have a proper steering mechanism, there is no direct control to the yaw direction for the robot. However, we can control the robot's heading direction by letting the robot to lean and precess until the desired direction is reached.

Although these special features bring a number of challenging problems in a control point of view, the high dynamic stability and maneuverability of Gyrover motivate us to have a further study on the robot, and to develop a complete control architecture for this system.

Chapter 3

Learning Control

Due to the complexity of the system, it is difficult for us to work out a 'complete' analytical model of it. Therefore, in this chapter, we propose using a machine learning algorithum, Cascade Neural Network (CNN) with node-decoupled extended Kalman Filtering (NDEKF), to model the robot's behaviors from human control strategy (HCS).

3.1 Motivation

Gyrover is a single track mobile vehicle which is inherently unstable in the lateral direction. With the lack of a wide polygon of support (single-point contact with the ground), Gyrover has a very bad static stability, even it has equipped with an internal gyroscope spinning at a high rate. The thin pneumatic tire which wrapped around the robot makes it difficult to stand in a stationary position for a very long time, it will fall on the ground eventually. However, by tilting the internal gyroscope into different orientations, we can indirectly control the lean angle of the robot, which implies that it is possible for us to keep the robot to stay around into its upright position with a proper control method.

Previous researches of Gyrover have been focused on the dynamics and control, including the kinematic constraints and motion equations [2, 3, 4, 5, 7, 8, 9]. However, the robot concept brings a number of challenging problems in modeling and control because of the highly coupled dynamics, the nonholonomic constraints and the non-minimum phase behavior of the system. The proposed linear state feedback model in [5] only gurantees the local stability of the system. Moreover, the dynamic model derived has been based on many assumptions which may not be realistic.

In [7], a linear state feedback controller is developed for stabilizing the robot to any desired angle, however, this model only applied for the case when the robot reaches at the steady state. By putting the consideration of the swinging motion of the internal mechanism, the model is modified in [8]. Unfortunately, the models obtained above are based on the assumption of rolling without slipping condition, that is, the robot must be rolling perfectly on the ground. Therefore, these models are not applicable for the static situation. In the static situation, the coupling between the wheel and the flywheel becomes much more complicated, which makes us difficult to derive an analytical model by traditional control method.

On the other hand, humans are capable of mastering complex, and highly nonlinear control system, a typical example is car driving. For Gyrover control, humans are able to control the robot well if enough practices (trainings) are given. Thus, we intuitively come up with the idea of machine learning, a model-free approach to model this kind of human control strategy. This approach is suitable for Gyrover control for the following reasons:

- Gyrover is a complex system which is difficult for us to develop a complete dynamic model to represent the robot's behaviors by using traditional control

method.

- In a practical point of view, it is equally difficult to model the system precisely due to some unmodeled factors, such as friction. Friction is an important issue when we are dealing with the coupling between the wheel and the spinning flywheel.

- Although Gyrover is a complex system, humans can control the robot through a radio transmitter to perform various kind of tasks, they do not need to explicitly model a system in order to control it. Through interaction with the system and observation of the behaviors of the system, humans are able to "learn" how to control a system.

- The learning process is in fact a direct input-output mapping between the system sensory data and the actuation data. A controller is generated by using the training data while a human "teacher" controls the system until the synthesized controller can perform the same way as human.

3.2 Cascade Neural Network with Kalman filtering

The field of intelligent control has emerged from the field of classical control theory to deal with applications which are too complex for classical control approaches. In terms of complexity, human control strategy lies between low-level feedback control and high-level reasoning, and encompasses a wide range of useful physical tasks with a reasonably well-defined numeric input/output representation.

Here, we introduce a continuous learning architecture for modeling human control strategies based on neural network. Since most neural networks used today

rely on rigid, fixed architecture networks and/or with slow gradient desent-based training algorithms, which may not be a suitable method to model the complex, dynamic and nonlinear human control strategy. To counter these problems, a new neural network learning architecture is proposed in [11], which combines (1) flexible cascade neural networks, which dynamically adjust the size of the neural network as part of the learning process, and (2) node-decoupled extended Kalman Filtering (NDEKF), a faster converging alternative to backpropagation. This methodology has been proved which can efficiently model human control skills [13, 14] and human sensation [30].

First of all, let's discuss the architecture of cascade learning. In cascade learning, the network topology is not fixed prior to learning, hidden units are added to an initially minimal network one at a time. This not only free us from a prior choice of network architecture, but also allows new hidden units to assume variable activation functions. That is, each hidden unit's activation function no longer need to confine to just a sigmoidal nonlinearity. A priori assumption about the underlying functional form of the mapping we wish to learn are thus minimized. The whole training process is described below:

1. Initially, no hidden unit exists in the network, only direct input-output connections. These weights are trained first, to obtain a linear relationship, if any.

2. With no further significant decrease in the RMS error (e_{RMS}), a first hidden node will be introduced into the network from a pool of candidate units. These candidate units are trained independently and in parallel with different random initial weights by using the quickprop algorithm.

3. The best candidate unit will be selected and installed into the network if no more appreciable error reduction occurs, therefore, the first hidden node is produced.

4. Once the hidden unit is installed, all the input weights of the hidden unit will be frozen, while the weights to the output unit(s) is/are going to train again. This allows for a much faster convergence of the weights during training than a standard multi-layer feedforward network.

5. This process (from step 2 - step 4) is repeated until the e_{RMS} reduces sufficiently for the training set or the number of hidden units reaches a predefined maximum number.

Figure 3.1 illustrates, for example, how a two-input, single-output network with a bias unit grows with increasing number of hidden nodes.

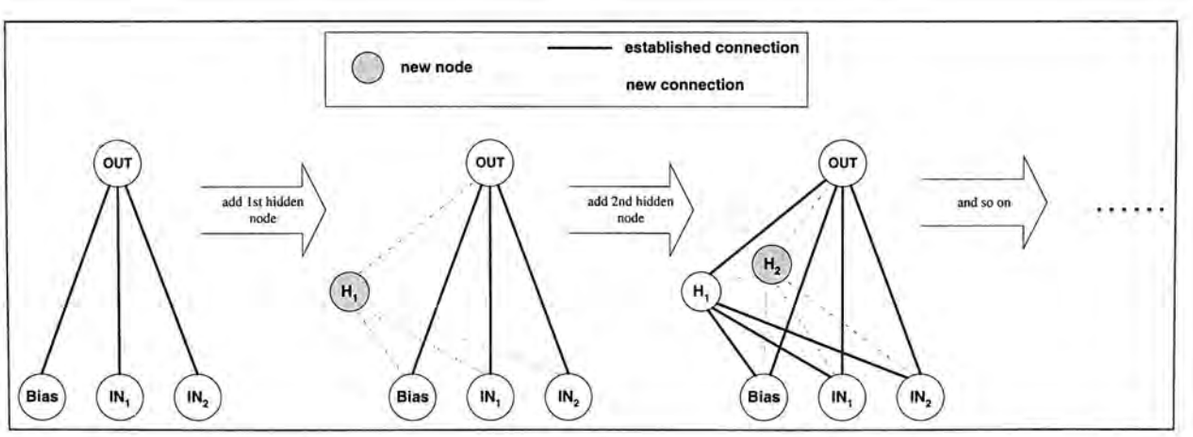


Figure 3.1: The cascade learning architecture.

A cascade neural network with n_{in} input units (including the bais unit), n_h hidden units, and n_{out}, has n_w connections (total number of weights) where,

$$n_w = n_{in}n_{out} + n_h(n_{in} + n_{out}) + (n_h - 1)n_h/2 \tag{3.1}$$

In fact, any multi-layer feedforward neural network with k hidden units arranged in m layers, fully connected between consecutive layers, is a special case of a cascade network with k hidden units with some of the weights equal to zero. Thus, this architecture relaxes a prior assumptions about the functional form of the model to be learnt by dynamically adjusting the network size. We can further relax these assumptions by allowing new hidden units to have different activation function. The kind of activation functions which reduces e_{RMS} most will be selected during the process, Sigmoid, Gaussian, and sinusoidal function of various frequency are some of the available types of activation functions we can choose.

While quickprop is an improvement over the standard backpropagation algorithm for adjusting the weights in the cascade network, it still requires many iterations until satisfactory convergence is reached. When combining cascade neural networks with node-decoupled extended Kalman filtering (NDEKF), [13] has shown that this methodology can solve the poor local minima problem, and that the resulting learning architecture substantially outperforms other neural network training paradigms in learning speed and/or error convergence for learning tasks important in control problems.

3.3 Learning architecture

Denote ω_k^i as the input-side weight vector of length n_w^i at iteration k, for $i \in \{0, 1, \ldots, n_o\}$, and,

$$n_w^i = \begin{cases} n_{in} + n_h - 1 & i = 0 \\ n_{in} + n_h & i \in \{1, \ldots, n_o\} \end{cases} \tag{3.2}$$

The NDEKF weight-update recursion is given by, (staring from equation (3.6)

to (3.9), {}'s, ()'s and []'s evaluate to scalars, vectors and matrics respectively)

$$\omega_{k+1}^i = \omega_k^i + \{(\psi_k^i)^T(A_k\xi_k)\}\phi_k^i \tag{3.3}$$

where ξ_k is the n_o-dimensional error vector for the current training pattern, ψ_k^i is the n_o-dimensional vector of partial derivatives of the network's output unit signals with respect to the ith unit's net input, and,

$$\begin{aligned}
\phi_k^i &= P_k^i\zeta_k^i && (3.4)\\
A_k &= \left[I + \sum_{i=0}^{n_o}\{(\zeta_k^i)^T\phi_k^i\}[\psi_k^i(\psi_k^i)^T]\right]^{-1} && (3.5)\\
P_{k+1}^i &= P_k^i - \{(\psi_k^i)^T(A_k\psi_k^i)\}[\phi_k^i(\phi_k^i)^T] + \eta_Q I && (3.6)\\
P_0^i &= (1/\eta_P)I && (3.7)
\end{aligned}$$

where ζ_k^i is the n_w^i-dimensional input vector for the ith unit, and P_k^i is the $n_w^i \times n_w^i$ approximate conditional error covariance matrix for the ith unit. The parameter η_Q is introduced in (3.9) to avoid the singularity problems for error covariance matrices, throughout the training, we use $\eta_Q = 0.0001$ and $\eta_P = 0.01$.

The vector ψ_k^i can be computed in this way: let O_i be the value of the ith output node, Γ_O be its corresponding activation function, net_{Oi} be its net activation, Γ_H be the activation function for the current hidden unit being trained, and net_H be its net activation. We have,

$$\begin{aligned}
\frac{\partial O_i}{\partial net_{Oj}} &= 0, \forall i \neq j && (3.8)\\
\frac{\partial O_i}{\partial net_{Oi}} &= \Gamma'_O(net_{Oi}), i \in \{1, \ldots, n_o\} && (3.9)\\
\frac{\partial O_i}{\partial net_H} &= w_{Hi} \cdot \Gamma'_O(net_{Oi}) \cdot \Gamma'_H(net_H) && (3.10)
\end{aligned}$$

where w_{Hi} is the weight connecting the current hidden node to the ith output node.

3.4 Input space

The cascade neural network architecture only offers a static mapping between the input and output. In fact, human control strategy is dynamic, we must map the dynamics system onto a static map. In general, we can approximate a dynamic system through a difference equation [13]:

$$\bar{u}(t+1) = \Gamma[\bar{u}(t), \bar{u}(t-1), \ldots, \bar{u}(t-n_u+1), \bar{x}, \bar{x}(t-1), \ldots, \bar{x}(t-n_x+1), \bar{z}(t)] \quad (3.11)$$

where $\Gamma(\cdot)$ is a mapping between a dynamic system onto a static one, $\bar{u}(t)$ is the control vector, $\bar{x}(t)$ is the system state vector, and $\bar{z}(t)$ is a vector describing the external environment at time t. Since vision system is not available on the current Gyrover prototype yet, the above equation is reduced to:

$$\bar{u}(t+1) = \Gamma[\bar{u}(t), \bar{u}(t-1), \ldots, \bar{u}(t-n_u+1), \bar{x}, \bar{x}(t-1), \ldots, \bar{x}(t-n_x+1)] \quad (3.12)$$

The order of the dynamic system in (3.9) is given by the constant n_u and n_x, which may be infinite. Therefore, by providing enough time-delayed histories of the state and command vectors of a system, a static model is able to abstract a dynamic system. For Gyrover, the HCS model will require:

1. **current and previous state information (e.g. lean angle of the robot, tilt angle of the flywheel),**

   $$\bar{x} = \begin{bmatrix} \beta & \beta_a & \dot{\beta} & \dot{\gamma} & \dot{\alpha} & \dot{\beta}_a & \ddot{\beta} & \ddot{\gamma} & \ddot{\alpha} & \ddot{\beta}_a \end{bmatrix}^T$$

2. **previous human operator's control information,**

   $$\bar{u} = \begin{bmatrix} u_o & u_1 \end{bmatrix}^T$$

Let's denote the HCS model's input space for Gyrover as,

$$\{\beta^{n_1}, {\beta_a}^{n_2}, \dot{\beta}^{n_3}, \dot{\gamma}^{n_4}, \dot{\alpha}^{n_5}, {\dot{\beta}_a}^{n_6}, \ddot{\beta}^{n_7}, \ddot{\gamma}^{n_8}, \ddot{\alpha}^{n_9}, {\ddot{\beta}_a}^{n_{10}}, u_o^{n_{11}}, u_1^{n_{12}}\}, \quad n_i \geq 0, i \in \{1, 2, \ldots, 12\}, \tag{3.13}$$

n_i is the number of time-delayed histories of a particular input variable. The above expression can also represent as,

$$\Xi^{n_i} = [\Xi(t - n_i + 1) \ldots \Xi(t-1)\Xi(t)]^T \qquad \Xi \in \{\bar{x} \;\; \bar{u}\} \tag{3.14}$$

The total number of inputs n_{in} is given by,

$$n_{in} = \sum_{i=1}^{12} n_i \tag{3.15}$$

Ξ will be ommitted from equation (3.14) if $n_i = 0$. For instance, $\{\beta^3, \beta_a^3, u_1^5\}$ represents a model whose input space consists of three previous lean angle (β) and tilt angle (β_a) information, and together with five history tilt motor commands (u_1). For the sake of convenient, we will set $n_x = n_1 = n_2 = \ldots = n_{10}$, and $n_u = n_{11} = n_{12}$. Therefore,

$$\begin{aligned} \{\bar{x}^{n_x}, \bar{u}^{n_u}\} &= \{\beta^{n_x}, {\beta_a}^{n_x}, \dot{\beta}^{n_x}, \dot{\gamma}^{n_x}, \dot{\alpha}^{n_x}, {\dot{\beta}_a}^{n_x}, \\ &\quad \ddot{\beta}^{n_x}, \ddot{\gamma}^{n_x}, \ddot{\alpha}^{n_x}, {\ddot{\beta}_a}^{n_x}, u_o^{n_u}, u_1^{n_u}\} \end{aligned} \tag{3.16}$$

$$n_{in} = 10n_x + 2n_u \tag{3.17}$$

3.5 Model evaluation

The main advantage of modeling robot's behaviors by learning, is that no explicit physical model is required, however, this also presents its biggest weakness. Since a model is trained by the input-output relationship only, the lack of a scientific jusitification degrades the confidence that we can show in these learnt models. This

is especially true when the process we are going to model is dynamic and stochastic in nature, which is the case of human control strategy. For a dynamic process, errors may feed back into the model to produce outputs which are not characteristics of the original process or making the process to be unstable. For a stocastic process, a static error criterion such as RMS error, based on the difference between the training data and the predicted model outputs is inadequate to gauge the fidelity of a learnt model to the source process.

In general, for different models, the similarity between a dynamic human control trajectory and a model-generated one will vary continously, from completely dissimilar to nearly identical. Furthermore, one cannot expect exact trajectories for the system and the learnt model, even equivalent initial conditions are given. To effectively evaluate the learnt models, we introduce a stochastic similarity measure proposed in [12]. This method is based on Hidden Markov Model (HMM) analysis, which is a useful tool for comparing stochastic, dynamic and multi-dimensional trajectories.

Hidden Markov Model is a trainable statistical model, which consists of a set of n states, interconnected by probabilistic transitions, each of these states has some output probability distributions associated with it. A discrete HMM is completely defined by,

$$\lambda = \{A, B, \pi\} \tag{3.18}$$

where A is the probabilistic $n_s \times n_s$ state transition matrix, B is the $L \times n_s$ output probability matrix with L discrete output symbols $l \in \{1, 2, \ldots, L\}$, and π is the n-length initial state probability distribution vector for HMM. Two HMMs (λ_1 and

λ_2) are said to be equivalent if and only if,

$$P(O|\lambda_1) = P(O|\lambda_2), \forall O \tag{3.19}$$

We prefer discrete HMMs than continuous or semi-continuous HMMs, because they are relatively simple in computation and less sensitive to initial random parameter settings. However, the human control trajectories we are going to measure are continuous and real-valued functions, in order to make use of the discrete HMMs, we must convert the data sets into sequences of discrete symbols O_n by the following procedures:

1. Normalization
2. Spectral conversion
3. Power Spectral Density (PSD) estimation
4. Vector quantization

The purpose of step (1) - (3) is to extract some meaningful feature vectors V for the vector quantizer. In step (4), the feature vectors V are converted to L discrete symbols, where L is the number of output observables in our HMMs.

In general, assume that we are going to compare the obervation sequences ($\bar{O}_1$ and $\bar{O}_2$) from two stochastic processes (Γ_1 and Γ_2). The probability of the observation sequences $\bar{O}_i$ given the HMM model λ_j, is given by [12],

$$P_{ij} = P(\bar{O}_i|\lambda_j)^{1/\bar{T}_i}, \qquad i,j \in \{1,2\} \tag{3.20}$$

where the above equation is being normalized with respect to the total numbers of symbols $\bar{T}_i$.

The similarity measure σ between $\bar{O}_1$ and $\bar{O}_2$ is,

$$\sigma(\bar{O}_1, \bar{O}_2) = \sqrt{\frac{P_{12}P_{21}}{P_{11}P_{22}}} \tag{3.21}$$

Figure 3.2 illustrates the overall approach to evaluate the similarity between two observation sequences. The HMMs are trained by each observation sequence first, then we cross-evaluate each observation sequence on the other HMM. Based on the four normalized probabilies, the similarity measure σ can be obtained.

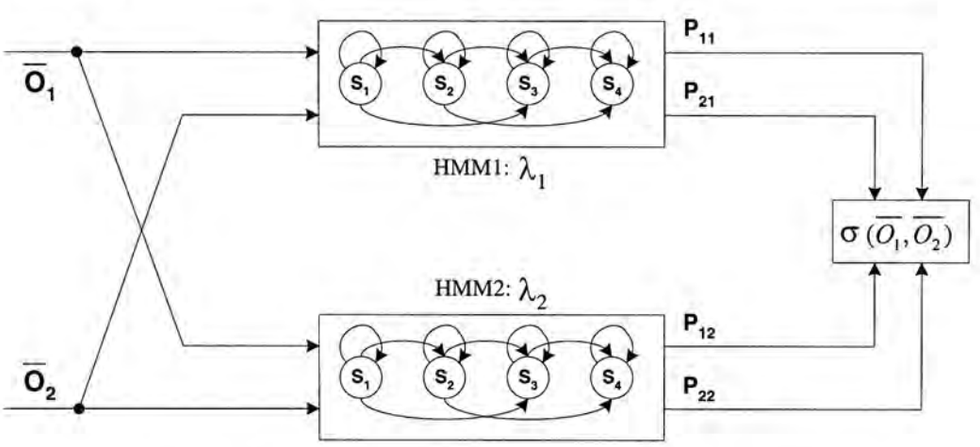
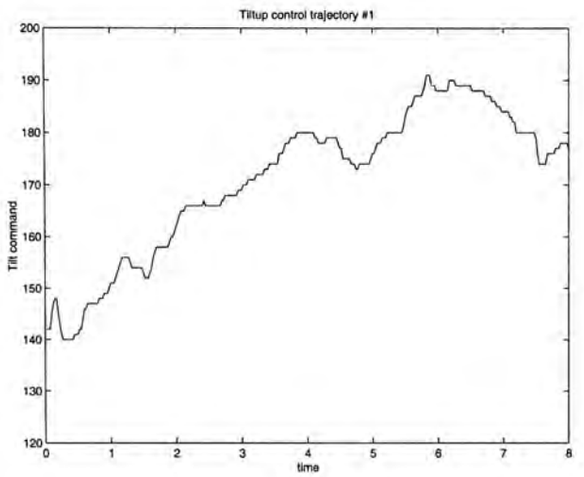


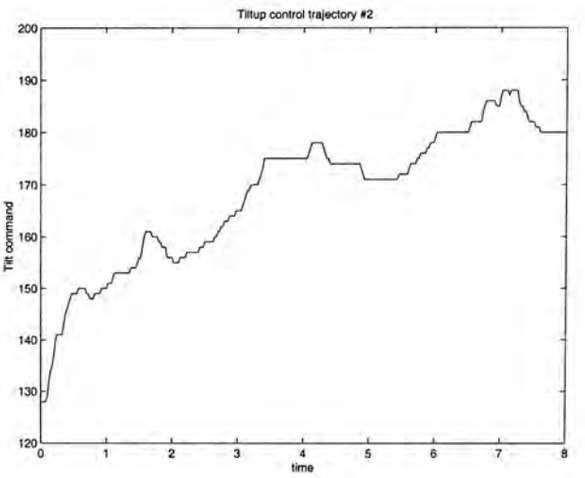
Figure 3.2: Similarity measure between $\bar{O}_1$ and $\bar{O}_2$.

Here, we demonstrate an example of how this similarity measure works. Figure 3.3 shows four Gyrover control trajectories. Figure 3.3(a) and 3.3(b) correspond to the tiltup motion control, while Figure 3.3(c) and 3.3(d) correspond to the lateral stabilization control of Gyrover. We applied the HMM similarity measure across these four trajectories, we might expect that the trajectories of the same motion should have a relatively high similarity, for any two trajectories which generated from different kinds of motion should have a low similarity value. We summarize the results in Table 3.1.

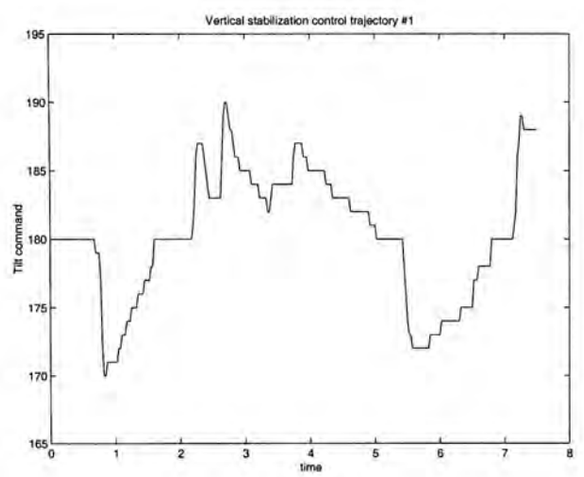
From the Table 3.1, it is clear that this similarity measure can accurately classify dynamic control trajectories from the same type of motion, while discirminating



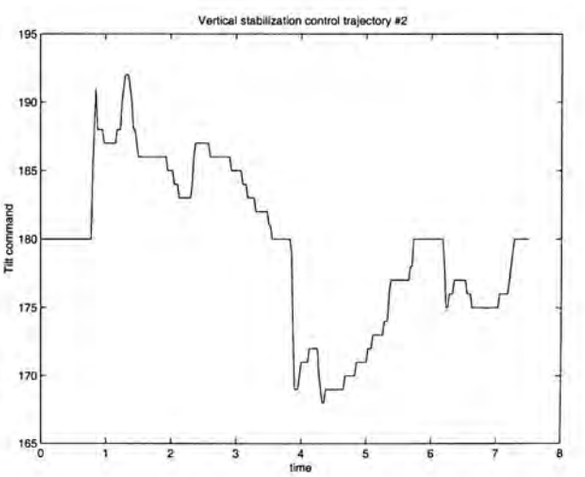
(a) Tiltup 1



(b) Tiltup 2



(c) Vertical balancing 1



(d) Vertical balancing 2

Figure 3.3: Control data for different motions.

	Tiltup #1	Tiltup #2	Vertical stab. #1	Vertical stab. #2
Tiltup #1	1.000	0.6306	0.0524	0.1234
Tiltup #2	0.6306	1.000	0.0615	0.0380
Vertical stab. #1	0.0524	0.0615	1.000	0.4994
Vertical stab. #2	0.1234	0.0380	0.4994	1.000

Table 3.1: Similarity measures bewteen different control trajectories.

those from different motions by giving a low similarity value. This similarity measure can be applied towards validating a learned model's fidelity to its training data, by comparing the model's dynamic trajectories in the feedback loop to the human's dynamic control trajectories.

3.6 Training procedures

Fist of all, we have made two assumptions for the training data provided for the learning process:

1. **Reliable training set.** Since learning is a kind of high-level, model free "teaching by showing" approach, the stability or robustness of the learnt model is heavily depended upon the operating skills of a "human teacher", in order to provide reliable and stable control. Therefore, throughout the teaching process, we assume that the operator is skillful and experienced enough to master the robot. That is, the training data can fully reflect the skills in a particular robot behavior. Besides the quality of the training data, the quantity of the data points is equally important. If the training set is in a larger scale, a more complete skill can be described.

2. **Injective mapping.** Another important issue is about the mappings between inputs and outputs in a static map. Figure 3.4 shows a human control strategy for the lateral balancing behavior, it is not difficult to figure out that the control of the flywheel is always switching (a very sharp change). That is, at a short moment ago, the command is positive, but in the next moment, the command will change into negative. Unfortunately, the switching problem causes very similar inputs to be mapped to a radically different outputs, which

is difficult for the cascade neural network to adapt, Figure 3.5. To ensure that there will be a correct mapping, enough time-delayed histories should be provided in the training data set. In our cascade network training, we will provide at least 20 history data ($n_i \geq 20$) to guarantee the injectiveness of the mapping.

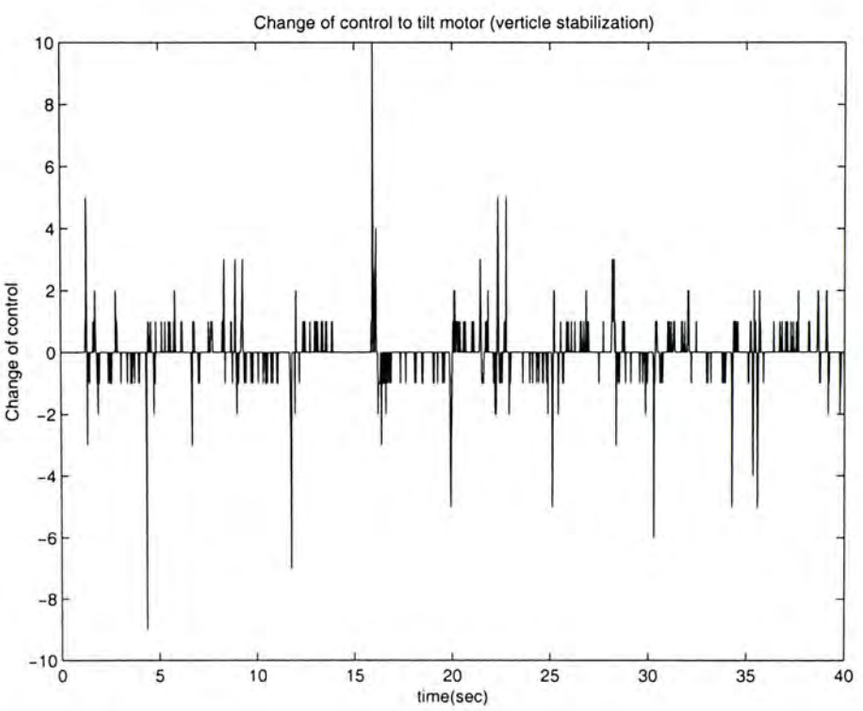


Figure 3.4: Switchings in human control of flywheel.

For each model, we process the training data as follows:

1. **Removal of irrelevant data**

   Let $[t, t + t_m]$ denotes an interval of time, in seconds, that a human operator has given an inappropriate command during the experiment. Then, we cut the data corresponding to time interval $[t - 1, t + t_m]$ from the training data. In other words, we not only remove the irrelevant data from the training set, but also the second data leading to the inappropriate command time interval.

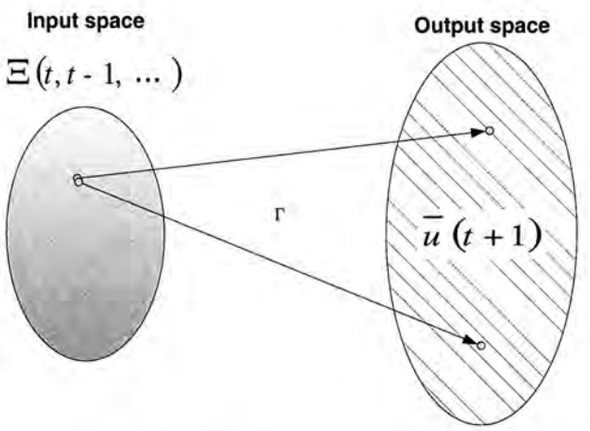


Figure 3.5: Similar inputs can be mapped to extreme different outputs if switching occurs.

This ensures that the cascade model does not learn control behaviors that are potentially destabilizing.

2. **Normalization**

   We normalize each input dimension of the training data, such that all the input in the training data falls inside the interval $[-1, 1]$.

3. **Generate time-shifted data**

   As mentioned in the previous section, we need to provide enough time-delayed values of each state and control variable such that the model is able to build necessary derivative dependencies between the inputs and outputs. In our cascade network training, we will provide 20 history data.

4. **Randomization**

   Finally, we randomize the input-output training vectors and select half for training, while reserving the other half for testing.

The sampling rate of the training data is 40Hz, typical training set will consist of approxiamtely 10,000 data points.

Chapter 4

Control Architecture

In this chapter, we will introduce the overall control architecture of Gyrover. Since the behavior-based control is widely used in mobile robot applications, we attempt to apply this control architecture into the Single Wheel Robot control system. Based on this concept, layers of control system are built to let the robot operate at increasing level of competence. By building this architecture, it gives us a clear picture to develop a complete control system for the robot.

4.1 Behavior-based approach

Behavior-based approaches have been established as a main alternative to conventional robot control in the recent years. Due to their modular architectures, these approaches provide high flexibility, while limiting complexity of individual modules. Each behavior in the system can be implemented and tested independently. Furthermore, they meet real-time requirements in a dynamic enviroment by creating a tight coupling between sensing and acting.

4.1.1 Concept and applications

A behavior-based approach has many advantages over traditional methods of controlling autonomous mobile robots. Traditional approaches decompose the overall problem into a set of functional units such as perception, world modeling, plan generation, etc. These functional units are linked sequentially that creating a linear datapath from sensory transducers to motor actuators, as shown in Figure 4.1.

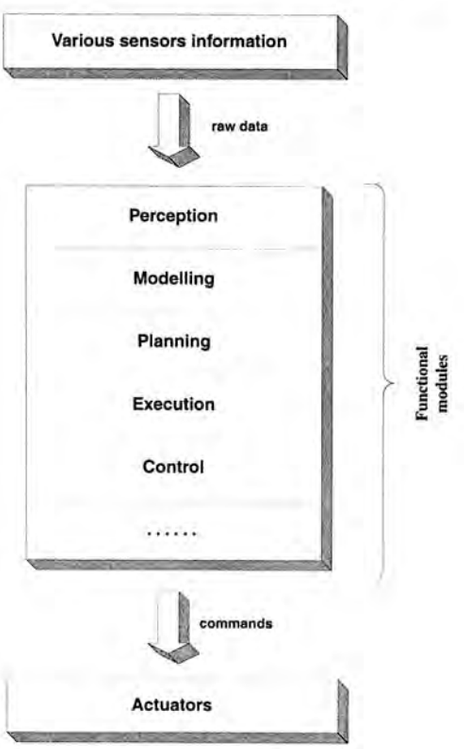


Figure 4.1: Conventional approach of a mobile robot control system.

That is, a robot first senses, perceives, and models its environment, and then it plans and acts in its environment. Since the world has plenty of information to acquire, this traditional method leads to information overload, which makes the robot incapable in functioning real time. Moreover, conventional methods assume the robot itself can construct accurate, global world models from the incoming sensory

information. The facts such as a rapidly changing world, limited processing power of the system, and inaccurate, incomplete sensor models make this assumption to be failed.

In contrast, a behavior-based approach solves the control problem in a parallel fashion, Figure 4.2. Each behavior, acting concurrently with other behaviors, only extracts the information required to complete a given task from the environment at a given time, which greatly avoid the information overload problem. This kind of division of labor method also eliminates the need for construction and maintenance of a global world model, which further reduces the computation load of the system.

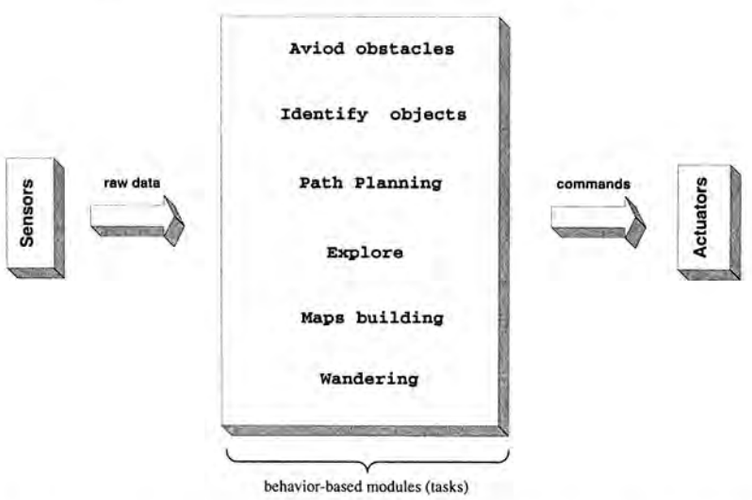


Figure 4.2: Behavior-based approach of a mobile robot control system.

Another advantage of the behavior-based approach is that it enables us to create layers of increasingly complex behaviors. The higher level behaviors can inhibit or modulate lower level behaviors. Therefore, a robot control system can be incrementally built with increasing capabilities, without losing low-level capabilities which are already created.

Behavior-based approach conveys significant contributions in the control of robotic

systems, a wide-range of robotic systems have applied this control architecture. The examples below illustrate the advantages of the applications of the hierarchical, behavior-based control in various kinds of autonomous systems:

- **Autonomous flying vehicle control**

  The University of Southern California Robotics Research Laboratory has developed an Autonomous Flying Vehicle-I (AFV-I) [18, 21]. A behavior-based control architecture was introduced for this autonomous flying vehicle. The behaviors of the robot are organized hierarchically, with low level, reflexive behaviors responsible for craft survival and high level behaviors responsible for tasks such as navigation and object location. The control system utilizes the sensors on AFV-I to make it to remain stable during the flight, navigation to a target, and to manipulate a physical object. The AFV-I had won the first-place in the International Aerial Robotics Competition in 1994.

- **Planetary autonomous robot control**

  In [16], [20] and [28], behavior-based control approach is applied to the field of planetary exploration. [16] presents a very small, legged robotic system, called the Mars Micro-Rover. The behavior-based architecture breaks down the Micro-Rover locomotion problem into many subtasks, from low level tasks (motor activities), medium level tasks (e.g. leg control) to higher level tasks (e.g. 'increase-ground-clearance').This mirco robot serves as a testbed to evaluate the performance potential of small legged robotic systems and their control architectures.

  Research groups of the Jet Propulsion Laboratory in California had implemented the behavior control algorithm in several microrover prototypes [20,

28]. The control systems of these microrovers integrate information from different sensors and encoders which report on the state of the articulation of the rover's suspension system and other mechanics, a homing beacon, a magnetic compass, and contact sensors. The robot is able to perform variety of useful tasks, such as soil sample collection, spectral imaging, and sample returns.

- **Multi-robot system**

  A multi-robot system is a system which consist of several autonomous robots working together to achieve a common goal. The most challenging problem of this system is how to effectively control a group of robots to perform a specific task and avoid collisions within the group. In [19], an approach is presented which is based on the master-slave type of control with dynamically selected 'master'. The implementation of the control system is a behavior-based, while the subsumption architecture is extended over a group of robots.

  A behavior-based formation control for multi-robot teams is presented in [24]. The formation behaviors are integrated with other navigation behaviors which enable a robotic team to reach navigation goals, to aviod hazards and remain in formation at the same time. The behaviors are implemented on robots in laboratory and aboard ummanned ground vehicles.

- **Mobile manipulation**

  A control architecture for mobile manipulation within a behavior-based framework, so called Mobile Manipulation Control Architecture (MMCA), is given in [26]. The control structure enables integration of the manipulator into a behavior-based control structure for the platform. This concept has implemented on a Puma560 arm which is mounted on a mobile platform.

Behavior-based control approach is well suited in Gyrover control for the following reasons:

- Multiple Goals: Since Gyrover has plenty of potential applications, it is necessary for the robot to perform multiple tasks simultaneously. It may require to reach a certain distance ahead while avoiding local obstacles. Moreover, often the relative importance of the goals will be context-dependent. For this kind of statical unstable vehicle, it is necessary to keep the robot remains stable in the lateral direction in all sense, whether the vehicle is in a static or dynamic status. The control system must be responsive to high priority but low level goals, e.g. lateral stability.

- Multiple Sensors: A number of on-board sensors have been installed on Gyrover to provide information about the state on the machine to the control computer. In reality, all sensors have an error component in their readings, and they will often give inconsistent readings. In a behavior-based architecture, not all sensors are required to feed into the central representation, only those with extreme reliability might be eligible to enter the central unit.

- Robustness: When some sensors on-board are failed, the robot should able to adapt and cope with the changes based on those remaining reliable sensors. The subsumption architecture can ensure that a degree of the behaviors is still functioning even some of the higher level modules has failed.

- Extensibility: Since more sensors and capabilities may be added into the system in the future, the existing control structure should be flexible enough for the builders to modify it.

4.1.2 Levels of competence

A level of competence is an informal specification of a desired class of behaviors for a robot over all enviroments it will encounter. A higher level of competence implies a more specific desired class of behaviors, each level of competence in fact includes a subset of each earlier level. Since each level defines a class of valid behaviors, it can be seen that higher levels provide additional constraints on that class. The key idea of levels of competence is that we can build layers of control system corresponding to each level of competence, by simply adding a new layer to an existing level, the capability of the existing set will be increased.

For instance, at the very beginning, we start by building a complete robot control system at the lowest level of competence. Since this layer represents the most basic task for the robot to execute (e.g. avoid hitting any obstacles), this layer is debugged thoroughly. Once this layer is completed, we never alter that system. Next, we build another control layer, which we call it the medium level control layer. This medium layer is able to examine the data from the lower level layer, and it also allows to inject data into the lower level which supresses the normal data flow. When the system is running, the lower layer continues to run unaware of the layer above it which may sometimes interferes with its data flow.

In such a way, additional layers can be added later, and the initial fundamental working system never needed to be altered. The same process is repeated in our design in order to achieve higher levels of competence for the system, as shown in Figure 4.3. This architecture is being well-known as a *subsumption architecture*. We will base on this idea to develop a behavior-based controller for Gyrover in the next section.

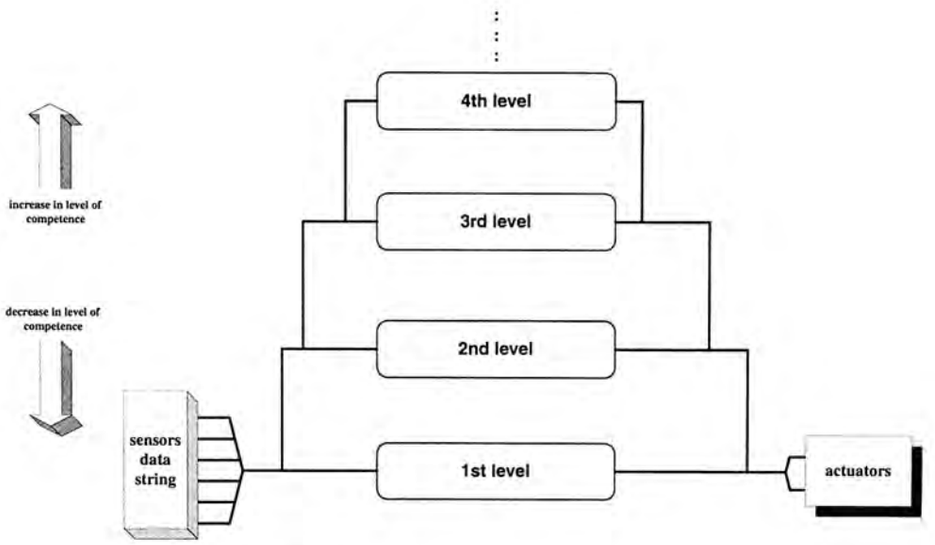


Figure 4.3: A subsumption architecture.

4.2 Behavior-based control of Gyrover: architecture

For building an autonomous control structure for Gyrover, we must first figure out the tasks which the robot can perform. By understanding the applications of the robot, we are able to list out some behaviors of the robot, and then we are going to design a behavior-based control architecture for Gyrover.

The Gyrover appears to be well suited in two classes of tasks: survey and transport. As a surveryor, Gyrover might carry a videocamera or other instruments for non-contact sensing, and survey broad regions at close range while travelling at high speed. Gyrover could be driven remotely, providing video data to seek out and explore sights for landing or construction, or paths for road construction. When the robot is equipped with some special sonsors on board, it is able to measure soil properties through the tire tread. As a transporter, Gyrover could carry equipment, materials or personnel. Because of its high dynamics stability, Gyrover can deliver tools or medical supplies rapidly. Moreover, the ability of fall recovery gives Gyrover

robustness and a high degree of survivability. The ability of Gyrover to travel on soft surfaces and water opens intriguing possibilities for an amphibious vehicle on earth.

Conventional autonomous mobile robots control usually focus on navigational problems such as goal seeking, path planning, obstacles aviodance and even speed control. Since they have a broad ploygon of support, they are very stable statically, and can tolerate large slopes without roll-over. However, due to the single-wheeled configuration together with the special steering and propulsion mechanism of Gyrover, the locomotion properties of this robot are slightly different from traditional quasi-static mobile robot. Although its slim profile can improve the maneuverability and can find obstacle-free paths more easily, the problem of low-speed stability is the one we need to tackle with in Gyrover control.

Gyrover consists a set of sensors (N, inputs) to perceive the environment, some actuators (U, outputs) to modify the enviorment or the robot's position, and together with a digital control system, which is equipped with some memory Z. From a mathmatical point of view, mobile robot control appears to be simple, theoretically, it is a mapping between the sensors n_i and the actuators u_i with a function f with respect to an internal memory state z_i, as the following equation denotes:

$$f : (n_i, z_i) \rightarrow (u_i, z_i') \qquad \text{or} \qquad (u_i, z_i') = f(n_i, z_i) \tag{4.1}$$

However, the above transformation is usually quite complex and highly nonlinear in real application. The dimension of sensor input N can be very high, but the dimension of actuator output U is typically small, or sometimes the internal state space dimension which is needed to perform a task is not even known. In general, we are unable to obtain a closed form representation for the function f.

By the way, we can reduce the complexity of the system by splitting the domain and dividing the problem into several sub-tasks (behaviors). Therefore, the problem becomes:

$$(u, z') = \begin{cases} f_1(n, z) & \text{if } (n, z) \text{ is in } B_1 \\ f_2(n, z) & \text{if } (n, z) \text{ is in } B_2 \\ \cdots & \\ f_n(n, z) & \text{if } (n, z) \text{ is in } B_n \end{cases} \tag{4.2}$$

where B_i represents a specific behavior of the robot.

The above expression can be further expressed as:

$$(u, z') = f_1(n, z) \cup f_2(n, z) \cup \cdots \cup f_n(n, z) \tag{4.3}$$

In fact, the sensor input, the actuator output and the amount of internal memory need not to be the same for each fucntion f_i, we have:

$$(u, z') = f_1(n_1, z_1) \cup f_2(n_2, z_2) \cup \cdots \cup f_n(n_n, z_n) \tag{4.4}$$

Therefore, each f_i is responsible for a mapping between sensors and actuators in a specific behavior subset. The number of sensors required in each f_i is not necessary the same as the others, which avoid data overflow for the system. Equation (4.4) is already the idea of a behavior-based control architecture.

Based on the discussions in the previous sections, we are able to build a prilimilary structure for Gyrover autonomous control. An overview of the control architecture is shown in Figure 4.4.

At the lowest level, the behaviors (reflex behaviors) implement very tight reflex loops. The task of each individual loop is very simple but essential. For instance,

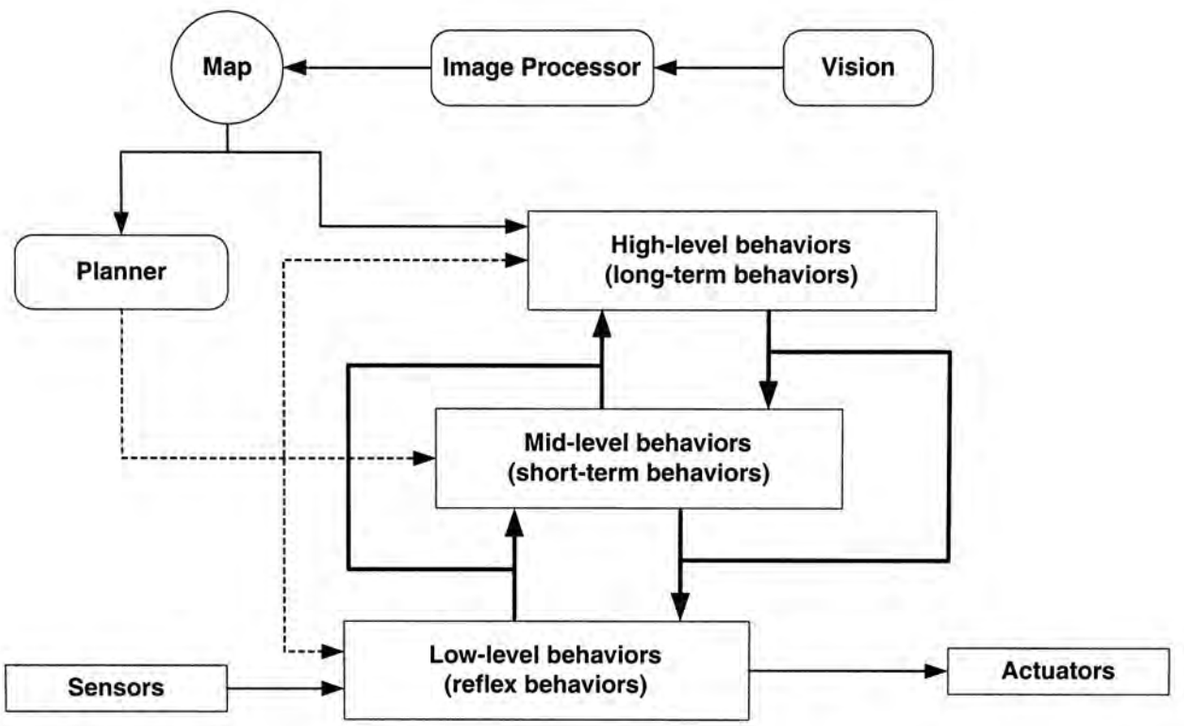


Figure 4.4: The overall control architecture.

the tiltup module responsible to tiltup the robot into the vertical upright position whenever the robot has fallen on the ground, which we hope the robot can perform this behavior even it cannot reach its higher level goal.

The medium level behaviors assume that the lower level behaviors are behaving with some degree of competence, they do not affect the outputs to the actuators directly but modulate the lower level behaviors. The behaviors in this level is also called the short-term behaviors, such as path tracking.

The high level behaviors is responsible for achieving some long term goals. The goal can be moving towards a target or searching for a specific target in a place. The planner, which is a much higher level module, is responsible for generating a set of subgoals to accomplish the entire task. This is done by activating the appropriate set of behaviors, and initiating the correct set of parameters.

At a very first step in building a behavior-based architecture for Gyrover control,

let we decompose the whole control task of Gyrover into several behaviors. We classify the behaviors of Gyrover into different levels: low, medium and high. The behaviors in the low level are (i) Lateral balancing, and (ii) Tiltup from the fall position. Medium level includes (iii) heading control, and (iv) obstacle avoidance. Behaviors such as (v) path planning and (vi) path tracking, are consider as high level behaviors.

In this way, based on the framework in Figure 4.4, we develop a behavior-based control structure for Gyrover. In Figure 4.5, most of the individual behaviors are shown, as well as the primary informational links.

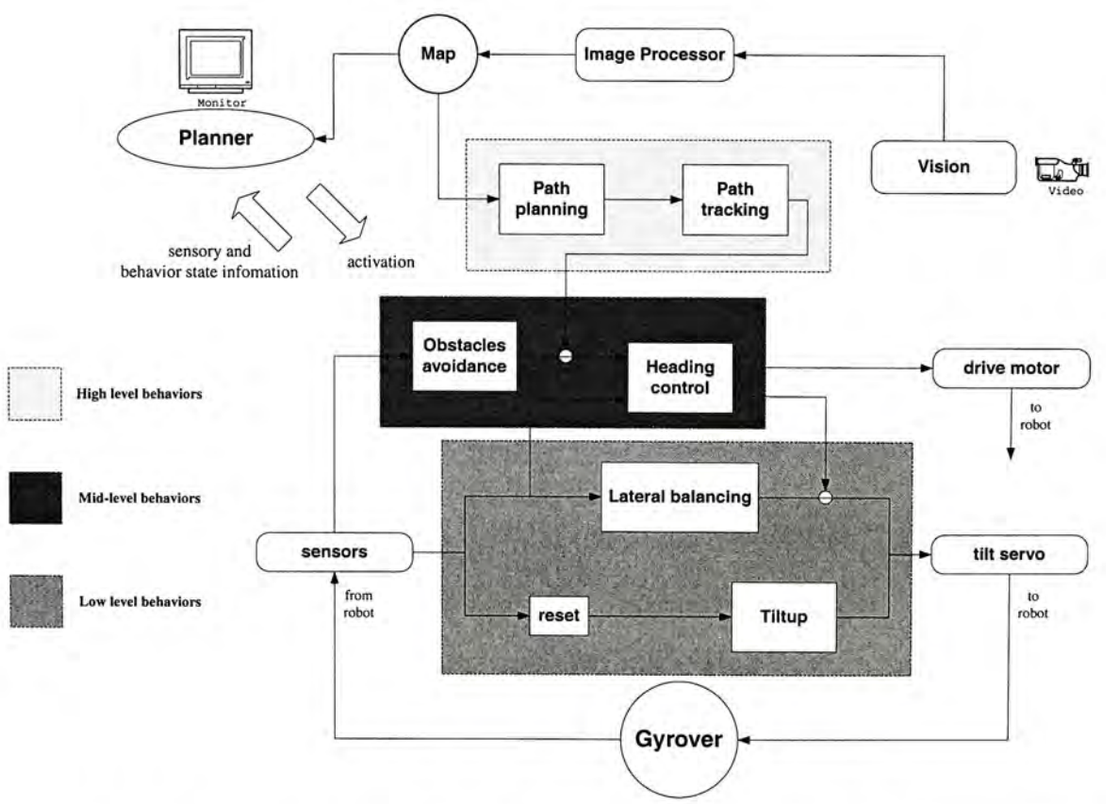


Figure 4.5: A detailed structure of the behavior connectivity in Gyrover control.

4.3 Behavior-based control of Gyrover: case studies

In order to develop an autonomous control scheme for Gyrover, we must deal with its lateral instablility problem, especially when the robot is in a static position (i.e. the robot does not roll). Recall the behavior-based control architecture we have developed in the previous section, we pick out the low-level behaviors module from the structure for further discussions, as shown in Figure 4.6.

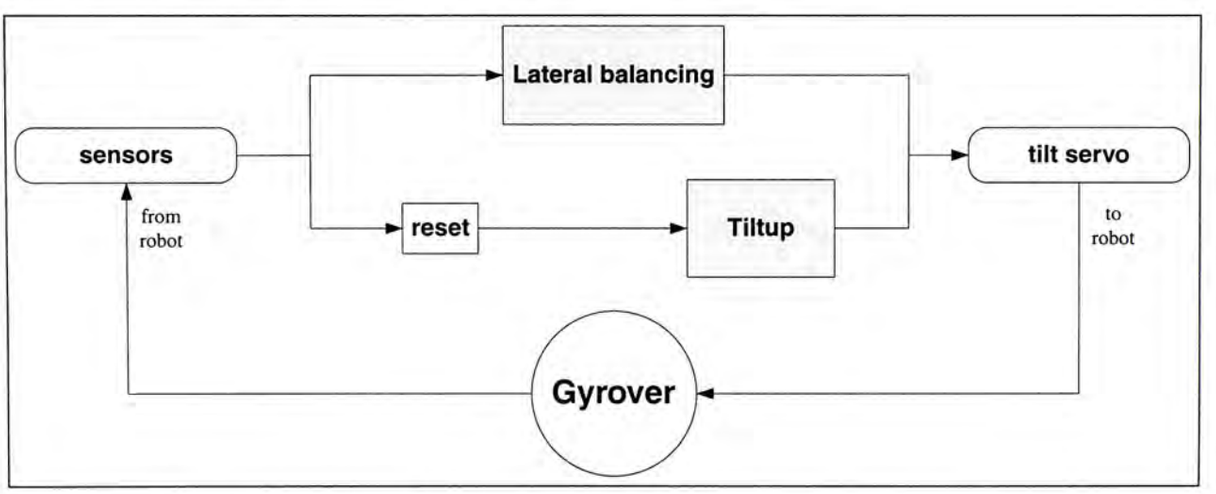


Figure 4.6: The low-level behaviors layer in the overall control architecture.

The shaded blocks in Figure 4.6 are the behaviors we desire the robot to perform in the first level of competence within the subsumption architecture, (i) Lateral balancing, and (ii) Tiltup motion. Therefore, if we can model these two behaviors, the statically unstable problem could be solved for Gyrover.

Humans are able to control the robot to perform complicated motions which are difficult to model in a mathmatical point of view. Therefore, we propose to approximate this human control capability using a "teaching by showing" approach [13, 14, 15].

4.3.1 Vertical balancing

Similar to the single track vechiles, Gyrover is inherently unstable in the lateral direction. The robot can easily fall down especially when its rolling speed is low or even it is not rolling. Fortunately, by tilting the internal flywheel, the coupling effect at yaw and row direction can somehow stabilize the robot in the vertical position. Therefore, we are seeking some control method to stabilize the robot in order to keep $\beta \approx 90^o$ for low speed as well as high speed operations. In Figure 4.7, under the control of human operator, the robot is able to stay roughly at 90^o in a 50 seconds experiment.

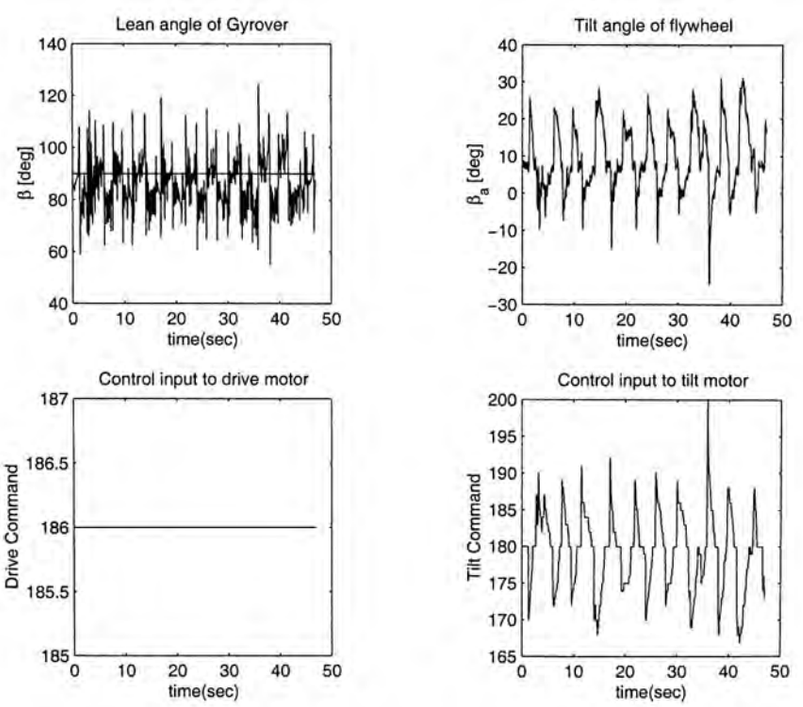


Figure 4.7: Lateral balancing at the vertical position (90^o) by human control.

	average lean angle $\bar{\beta}$	$DOF_{flywheel}$
Human control	89.32^o	0.9600

Table 4.1: Performance of human operator in verticale stabilization

Table 4.1 summarized the robot status throughout the experiment. The operator can control the robot to stay at around 90^o while reserve a high degree of freedom for the flywheel. Thus, this motion is selected as one of the behaviors we are going to "teach" the robot.

4.3.2 Tiltup motion

Tiltup motion is refered to the behavior that the robot recovers from the fall position ($\beta \approx 20^o$) back to the upright position ($\beta \approx 90^o$), which is an unique behavior of Gyrover over traditional multi-wheels mobile robots. In [14], a tiltup motion which is constituted by the control of the drive motor u_0 and tilt motor u_1 simultaneously is introduced. However, we found that the tiltup motion in [14] brings a number of problems in applications: (1) require a large space to perform this motion, (2) the final heading direction α is unpredictable, and (3) it takes a longer period of time to complete. Thus, we modified the previous tiltup motion by considering the control of flywheel only. Figure 4.8 shows the performance of the modified tiltup motion.

In Figure 4.8, the robot is orginally lying on the ground with lean angle at 20^o, 1 second later, the operator changed the orientation of the flywheel and the robot is back to its upright position a moment later. The drive motor command is kept constant at the 0 position implies the robot is not moving neither forward nor backward.The modified motion outstands the previous one for the following reasons:

- Takes shorter time to finish
- Not much space is needed because the robot can be tiltup at nearly the identical position
- Heading direction is predictable since the heading direction before and after

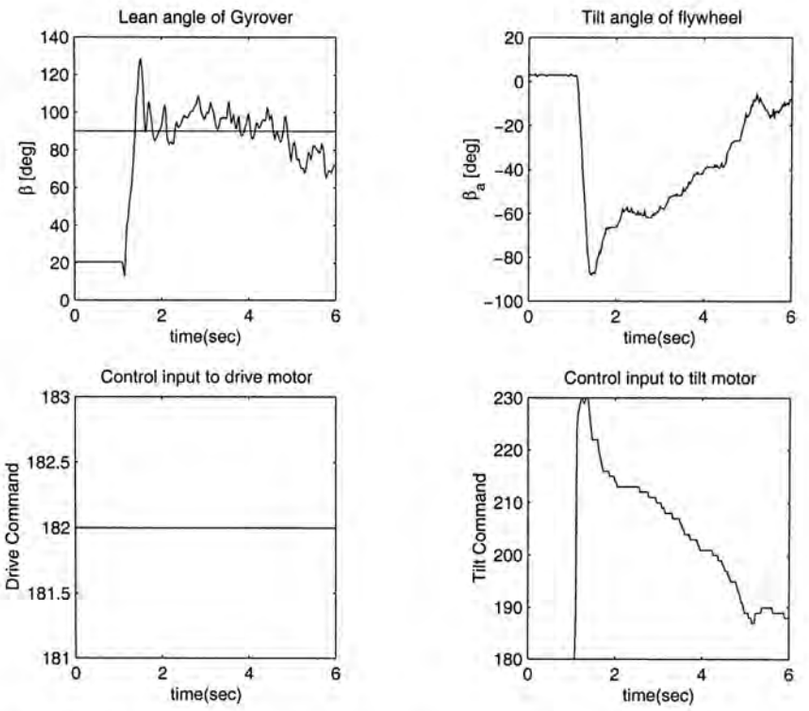


Figure 4.8: Modified tiltup motion by human control.

tiltup does not vary too much

- Internal pendulum motion is avoided

Therefore, besides the motion of lateral stabilization, the modified tiltup motion is another behavior which we are going to let the robot to learn from human.

4.4 Discussions

From the behavior-based architecture we obtained in the previous section, we can recognize that the entire control task is decomposed into many sub-tasks which located at different levels within the structure.

Low-level behaviors

As mentioned in the earlier section, Gyrover is inherently unstable and underactuate in its lateral direction, the stability is improved when the robot is equipped with an internal gyroscope spinning at a high rate. By controlling the orientations of the internal gyroscope, we are able to stabilize the robot into its upright position even the robot is not rolling. For the case when the robot has fallen onto the ground, it is able to tilt-up by itself. Therefore, the lateral balancing behavior and the tiltup motion constitute the basic level control of Gyrover.

This lowest layer of control makes sure that the robot can maintain its lateral stability in a static condition (when the wheel does not roll) and can recover from fall. Therefore, no matter the robot is rolling or not, once the control is activated, we suppose the robot will keep standing upright. This complete the first level of competence in the control structure.

Mid-level behaviors

The mid-level layer of control, when combined with the low-level layer, the robot can move around without hitting obstacles while it can still maintain its lateral stability, and will recover from the fall positions when the robot falls down. Besides the direct actuation of the drive motor in the heading control module, the behaviors in this layer only affect the system by modulating the low-level reflex behaviors. This was defined as second level of competence in this architecture.

If an obstacle is detected by the robot in a certain range, the Obstacle Avoidance module will generate a command to modify the robot's heading direction in the Heading control module, so that the robot will not get hit on the obstacle.

High-level behaviors

This level is meant to add an exploratory mode of behavior into the robot. The decisions made in this layer are some long-term goals relative to that of the former layers, for example, to find an obstacle-free path to reach a distance location from the current location. Although a map is necessary to cope with this module in order to generate a desired path within a region, vision is not the only way to generate such a map, other alternatives such as a GPS may also be used. The commands generated from this layer will also suppress the lower level module to accomplish the third level of competence in this system.

In addition, there is an external module to monitor the robot's actions, called the planner, appeared in the top left corner in Figure 4.5. The planner is responsible for producing the set of actions that achieve a certain goal for the robot. For each stage in the plan, the appropriate set of behaviors are activated. This unit can be an on-board unit or can be a tele-operating unit.

Although we are still in an early stage to complete the mid-level and high-level layers based on the current system we are using, it is worthwhile to develop such an architecture for us to build a fully-autonomous control system for Gyrover in the near future.

In summary, the behavior-based approach is suitable for Gyrover control for the following reasons :

- This control system is able to respond to high priority goals (e.g. path planning), while it can still servicing necessary low-level goals (e.g. the lateral stability)

- This *subsumption* architecture enables us to extend the whole system into a

more complete one if we have explored other tasks for the robot to perform in the future.

- If some of the modules in higher level are failed to work properly, the robot can still perform some low-level instinct behaviors.

- Numerous inertial sensors and a mirco-computer is begin built on board in the third prototype of Gyrover. If all the sensors data are fed in each of the sub-task controller, the computational time for each response will increase significantly.

- Since Gyrover is designed for general transportation, exploration, rescue or recreation. Individual layers can be working on individual goals concurrently.

This subsumption architecture leads us the idea of share control (semi-autonomous control) for Gyrover, which will be discussed in Chapter 6.

Chapter 5

Implementation of Learning Control

In this chapter, we show the implentation results of the CNN models trained in the previous chapter. First of all, we validate the CNN models we obtained by applying a Hidden Markov Model based similarity measure. Next, for the experimental implementations of the CNN models, we evaluate the performance of these models by observing the lean angle of the robot and the overall control on the flywheel. Later, we combined the two motions into a single motion. This combined motion ensures that the robot can be fully recovered from the fall position back and balanced at its upright position.

5.1 Validation

In this section, we will evaluate each of the model generated by the cascade learning algorithm for different behaviors of the robot, including lateral stabilization and tiltup motion. We apply the similarity measure mentioned in Section 3.2.4 to quantify the level of similarity between the original human control data and the model-generated trajectories through simulations. Since we do not have a physical

model for these kind of motion for Gyrover, our simulations are done by feeding the current and history state variables and control information into the cascade neural network, to see if it can generate similar control output in each time instant.

Basically, we have two motions to learn: (1) Lateral balancing ($i = 1$), and (2) Tiltup ($i = 2$). For each motion, we give three different set of data for the simulation. For notation convenience, let $X^{i,j}$, $i \in \{1, 2\}$, $j \in \{1, 2, 3\}$, denote the run of different motions i in trail $\#j$.

5.1.1 Vertical balancing

Figure 5.1, 5.3 and 5.5 show three different vertical balanced motion by human control. The graph on the left of each figure is the plot of lean angle data (β), while the right one plots the orientations of the flywheel (β_a). The corresponding human control data and CNN model control data for $X^{(1,1)}$, $X^{(1,2)}$ and $X^{(1,3)}$ are shown in Figure 5.2, 5.4 and 5.6 respectively. We perform the similarity measure between the human control and CNN model control trajectories for each motion, the results are summarized in Table 5.1. From the performance of this vertical balancing CNN model, we can observe that the model can generate similar control trajectories as human operator, with an average similarity value of 0.5940.

	similarity σ
$X^{(1,1)}$	0.5885
$X^{(1,2)}$	0.6235
$X^{(1,3)}$	0.5700
average	0.5940

Table 5.1: Similarity measures for vertical balanced control betweem human and CNN model

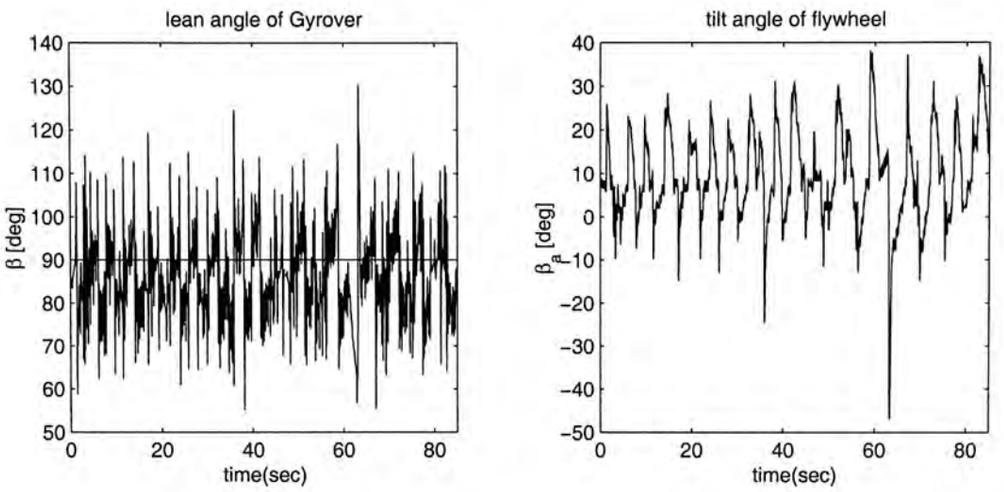


Figure 5.1: Vertical balanced motion by human control, $X^{(1,1)}$.

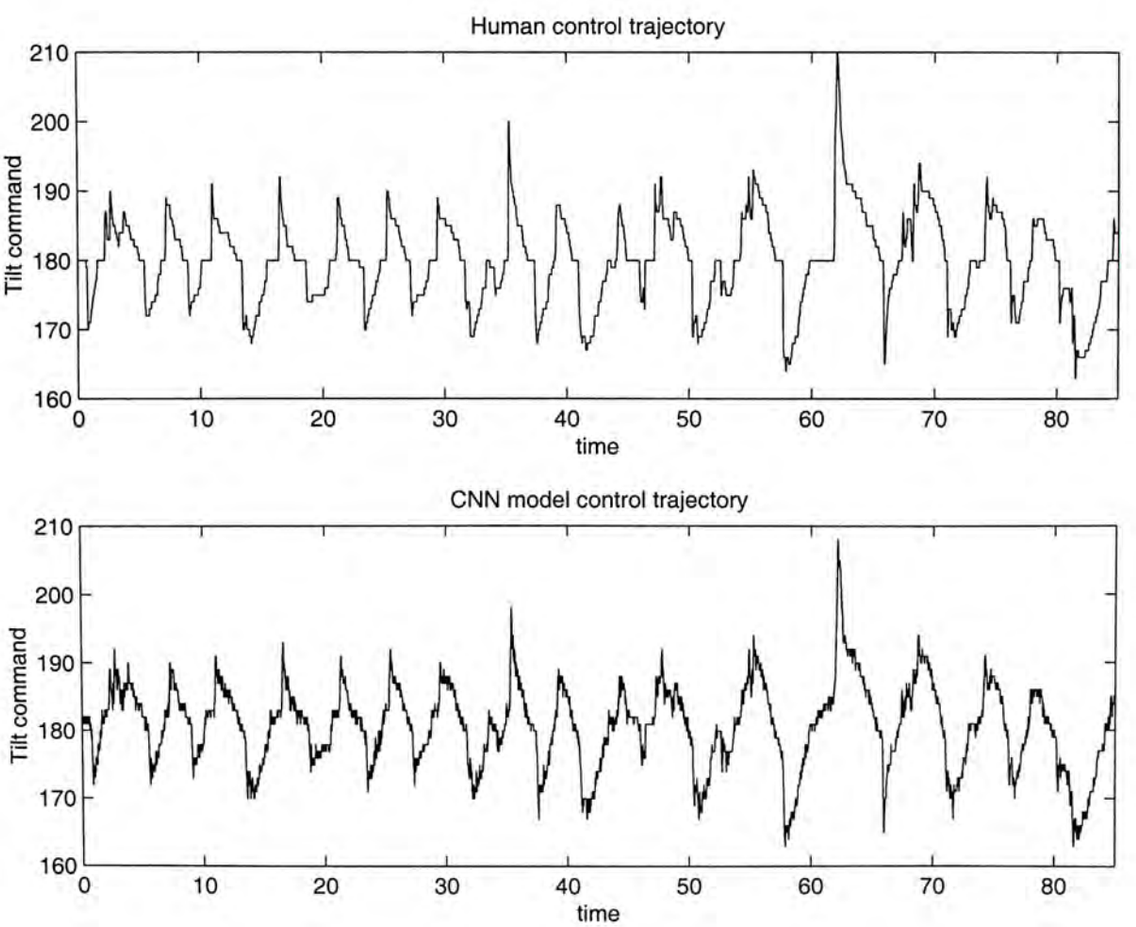


Figure 5.2: Control trajectories comparison for $X^{(1,1)}$.

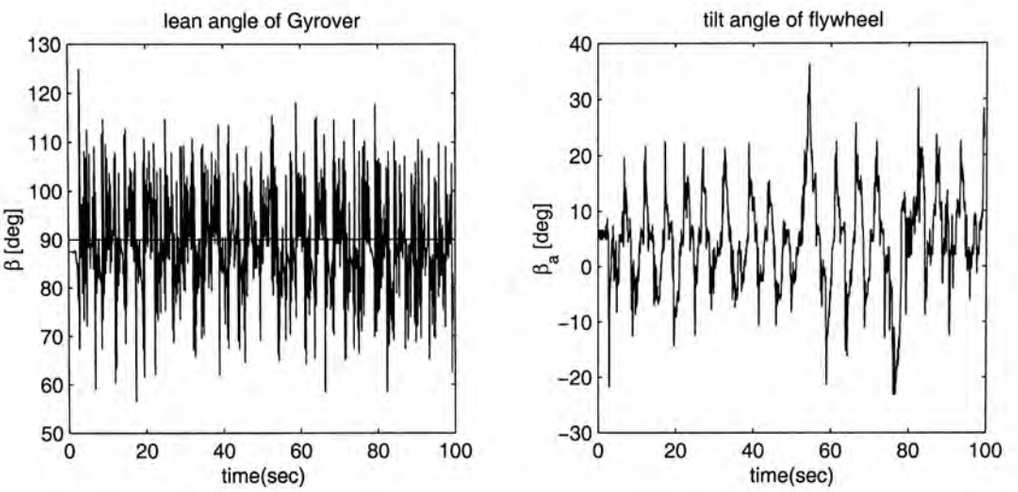


Figure 5.3: Vertical balanced motion by human control, $X^{(1,2)}$.

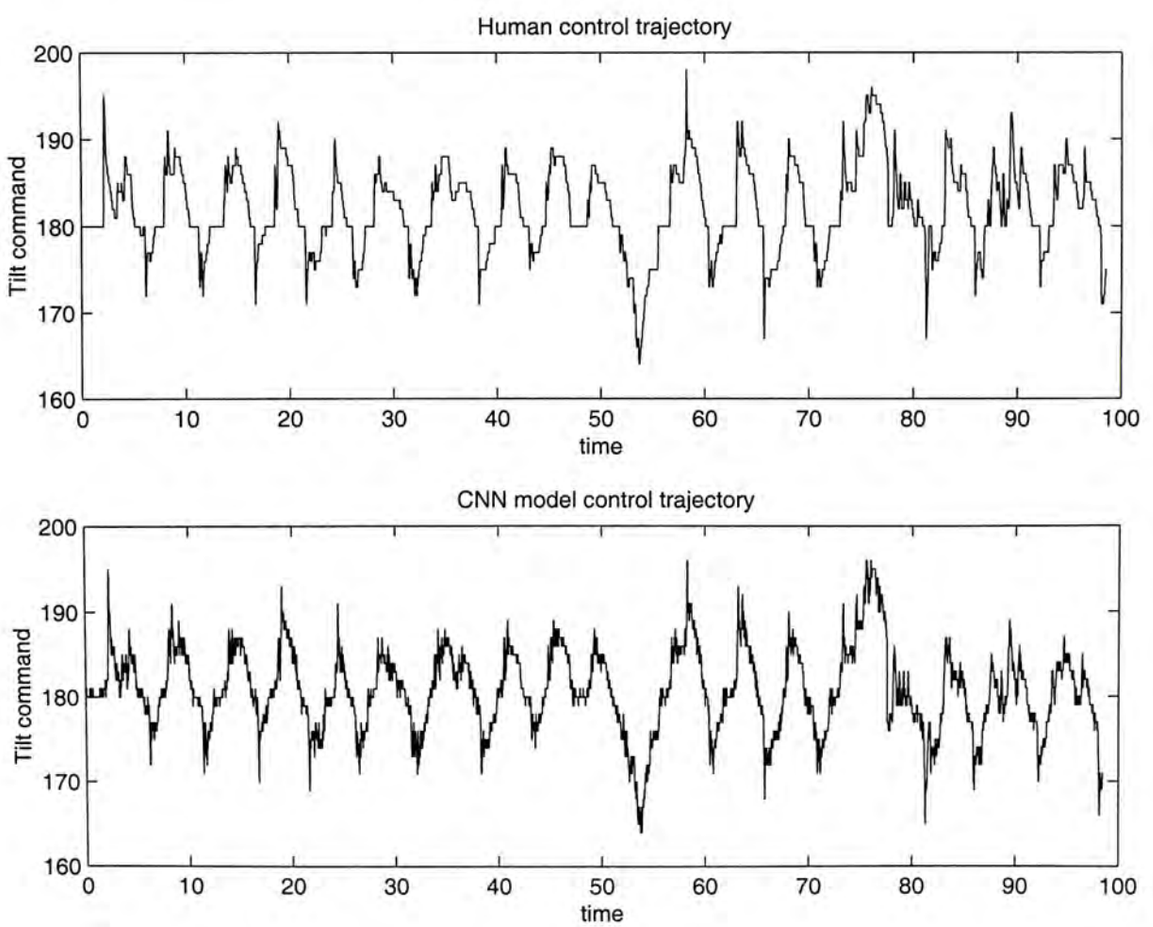


Figure 5.4: Control trajectories comparison for $X^{(1,2)}$.

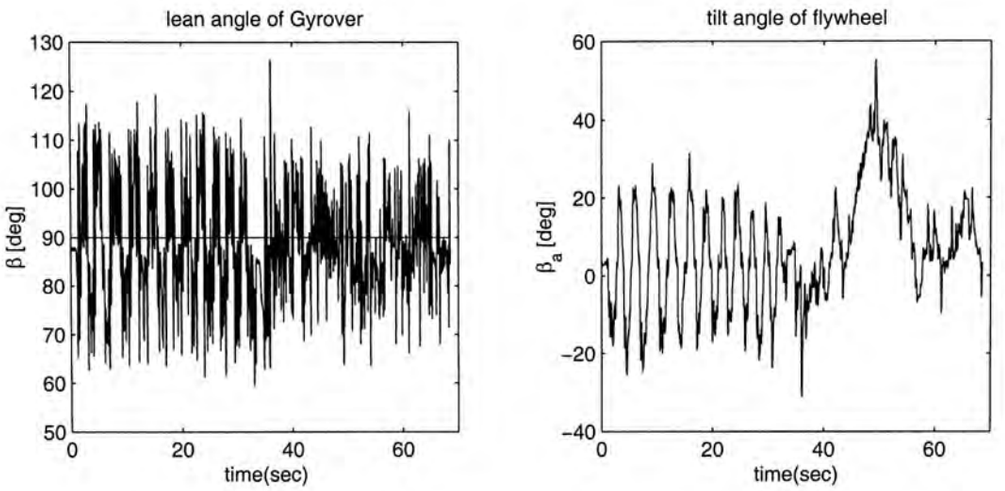


Figure 5.5: Vertical balanced motion by human control, $X^{(1,3)}$.

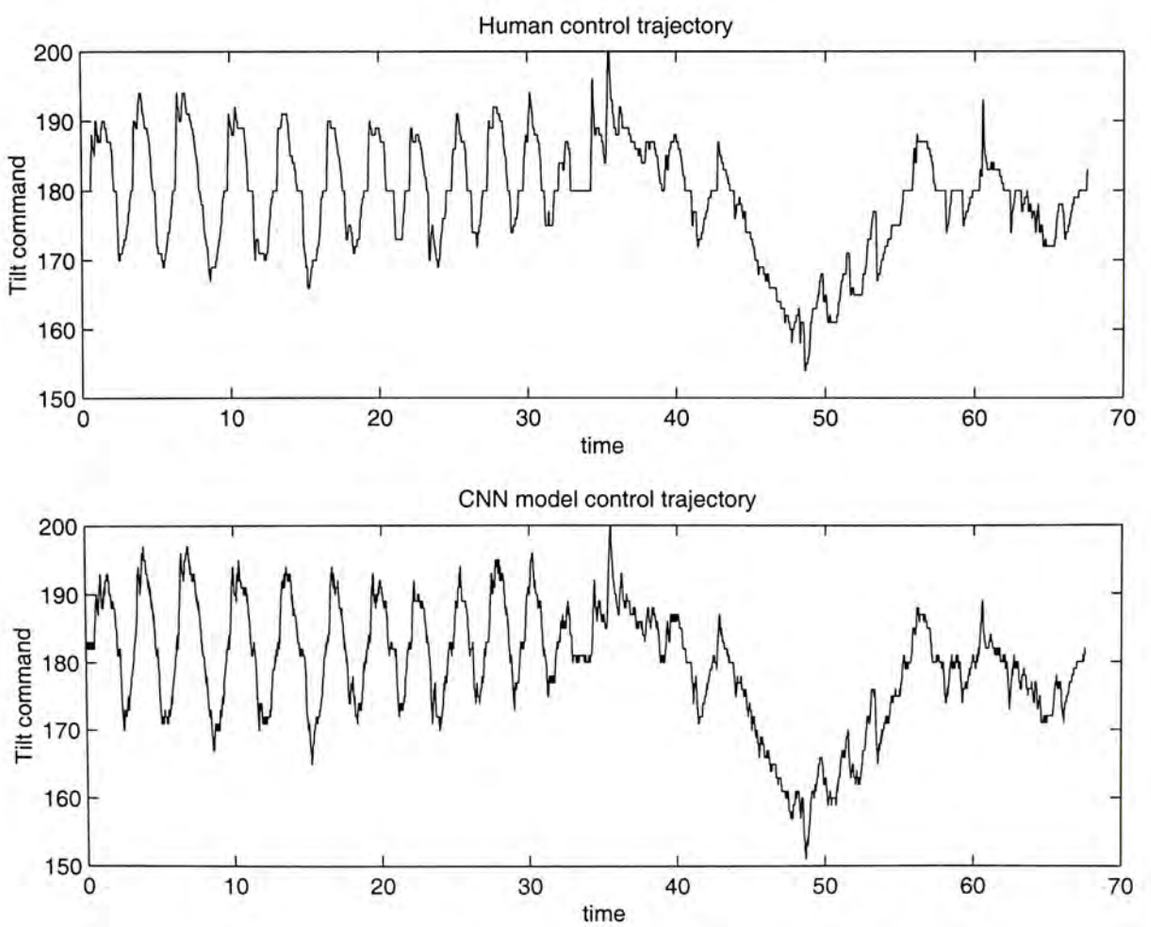


Figure 5.6: Control trajectories comparison for $X^{(1,3)}$.

5.1.2 Tilt-up motion

Figure 5.7, 5.9 and 5.11 show three different tiltup motion by human control. The corresponding human control data and CNN model control data for $X^{(2,1)}$, $X^{(2,2)}$ and $X^{(2,3)}$ are shown in Figure 5.8, 5.10 and 5.12 respectively. Again, we perform the similarity measure between the human control and CNN model control trajectories for each motion, the results are summarized in Table 5.2. The CNN model can also generate similar control trajectories as human operator, with an average similarity value of 0.7437.

	similarity σ
$X^{(2,1)}$	0.7896
$X^{(2,2)}$	0.7030
$X^{(2,3)}$	0.7386
average	0.7437

Table 5.2: Similarity measures for tiltup control betweem human and CNN model

5.1.3 Discussions

The simulations we have done in fact is the first step to validate the CNN models we obtained. By using the HMM similarity measure, we compare the human control trajectory with the control trajectory generate from the CNN model of a particular motion. If the similarity measure gives us a relatively high similarity value ($\sigma \geq 0.5$), which implies the particular CNN model can produce 'similar' control output as human control. From the simulation results of the lateral balancing and tiltup motion, we can verify that the CNN models for both motions are able to model the human control strategy. Later on, in the next chapter, we will further verify the models by experimental implementation.

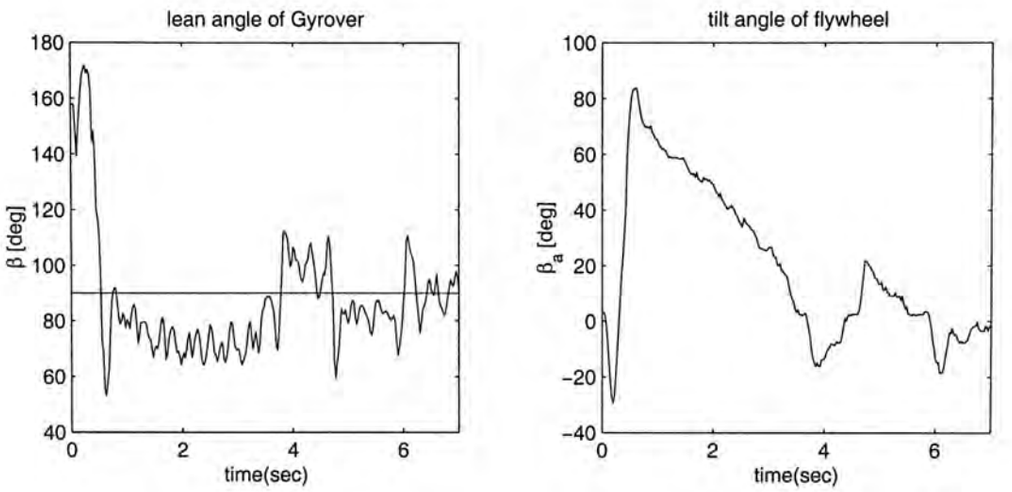


Figure 5.7: Tiltup motion by human control, $X^{(2,1)}$.

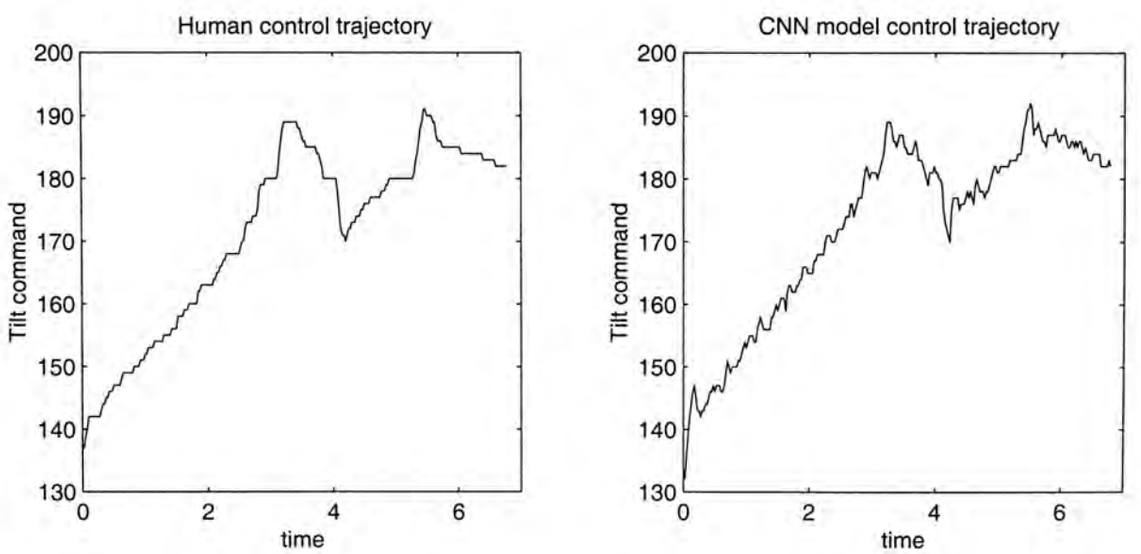


Figure 5.8: Control trajectories comparison for $X^{(2,1)}$.

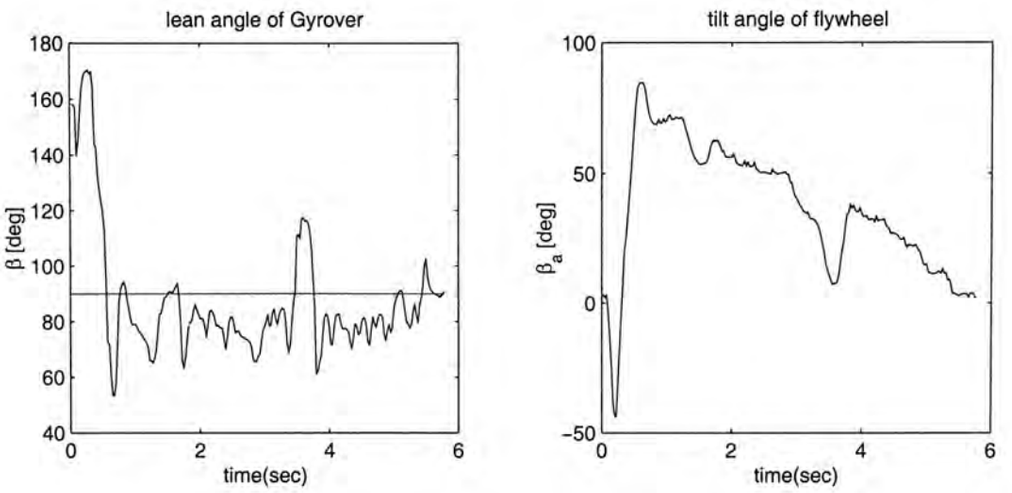


Figure 5.9: Tiltup motion by human control, $X^{(2,2)}$.

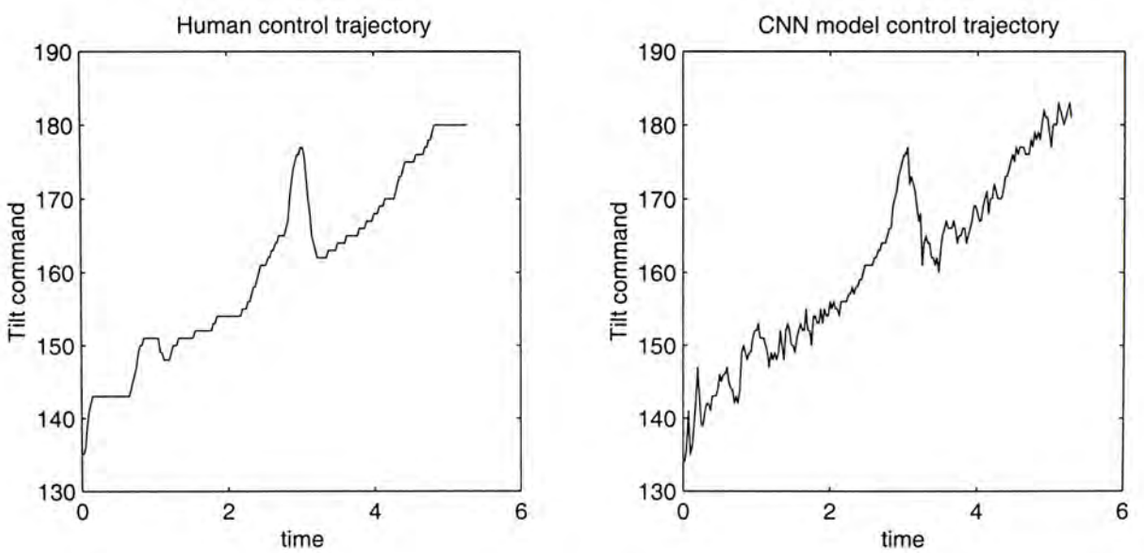


Figure 5.10: Control trajectories comparison for $X^{(2,2)}$.

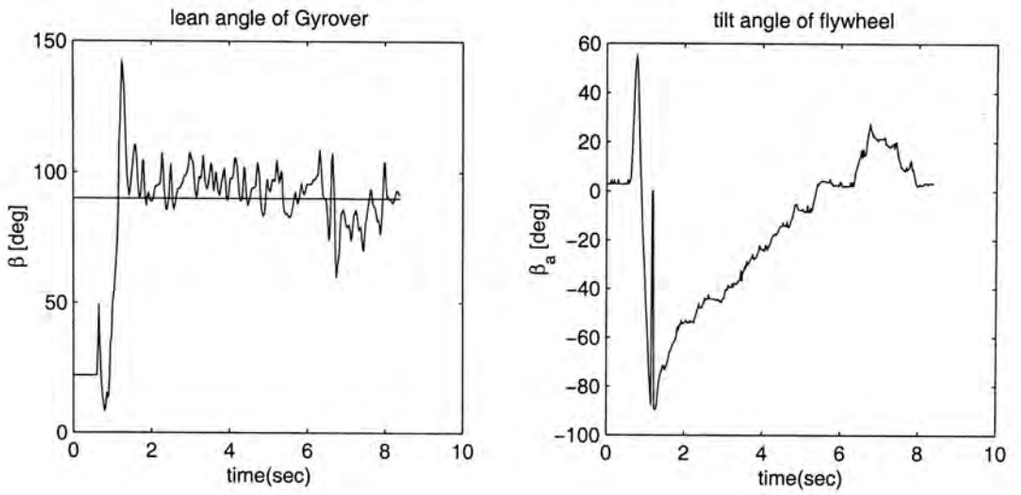


Figure 5.11: Tiltup motion by human control, $X^{(2,3)}$.

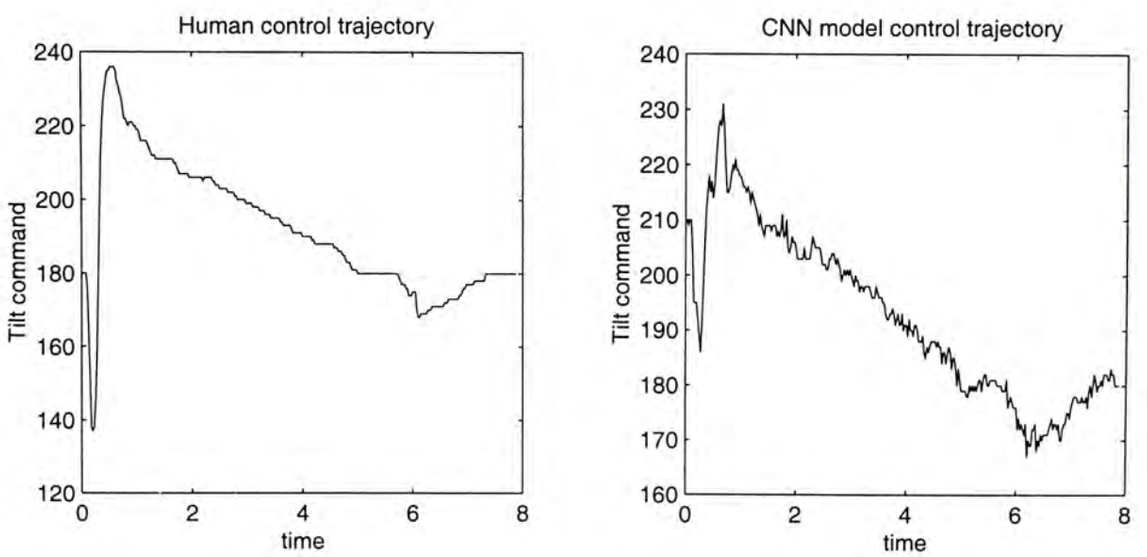


Figure 5.12: Control trajectories comparison for $X^{(2,3)}$.

5.2 Implementation

5.2.1 Vertical balanced motion

A number of experiments have been conducted to verify the CNN model for vertical balancing, Figure 5.13, 5.14 and 5.15 shows the implementation results. The human control strategy in balancing the robot at the vertical position is given in Figure

5.16. As mentioned in the pervious chapter, we evaluate the performance by the lean angle of the robot and the degree of freedom remains for the flywheel. We summarized the overall performance of both CNN model and human operator for the vertical stabilized motion in Table 5.3.

	average lean angle $\bar{\beta}$	$DOF_{flywheel}$
CNN control #1	90.24^o	0.9944
CNN control #2	88.11^o	0.8756
CNN control #3	87.57^o	0.8867
Human control	89.41^o	0.9600

Table 5.3: Performance measures for vertical balancing.

When compared with human control, the CNN model we obtained for vertical balancing behaves very similar to human. For the 3 different trails, the CNN model not only able to stablize the robot at around 90^o, but also reserved a high level of degree of freedom for the internal flywheel to oppose any motion that appears to make the robot to fall down.

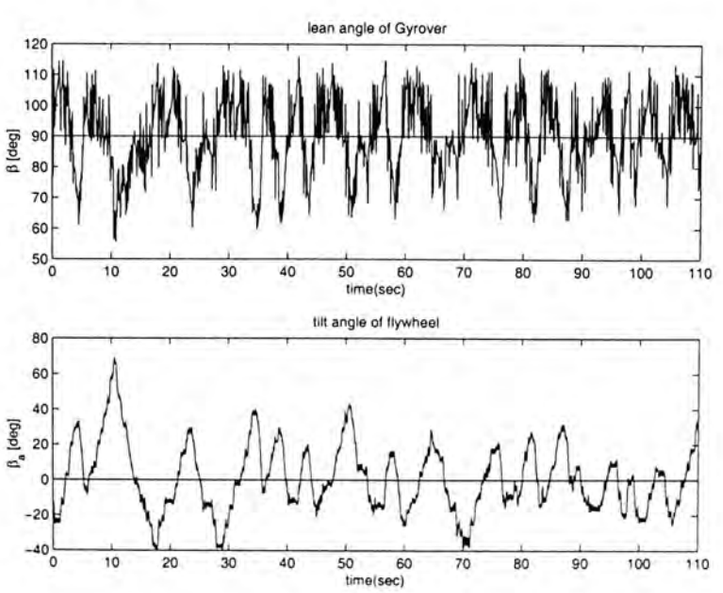


Figure 5.13: Vertical balancing by CNN model, trail #1.

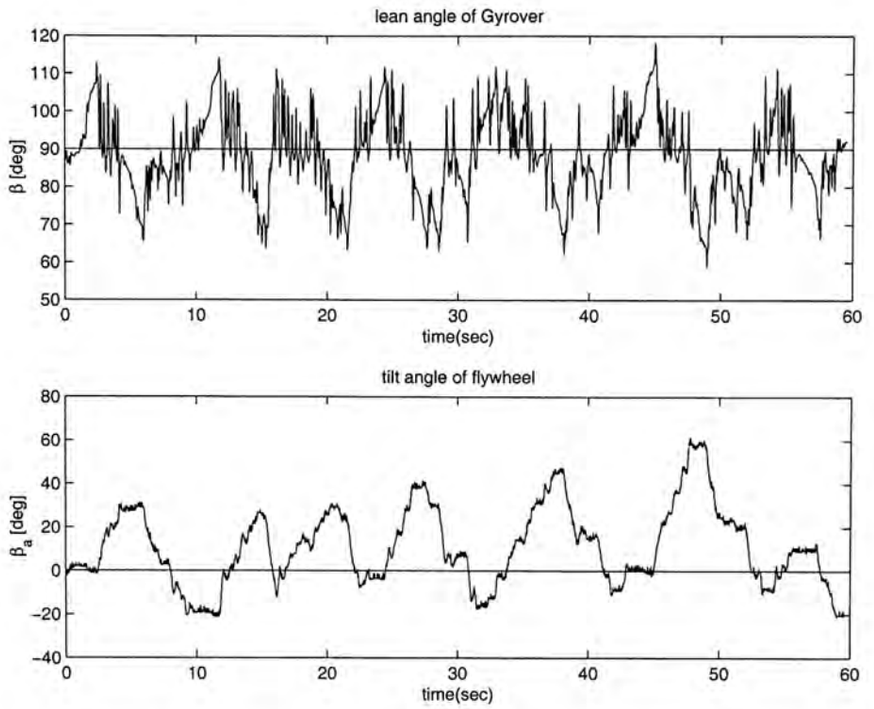


Figure 5.14: Vertical balancing by CNN model, trail #2.

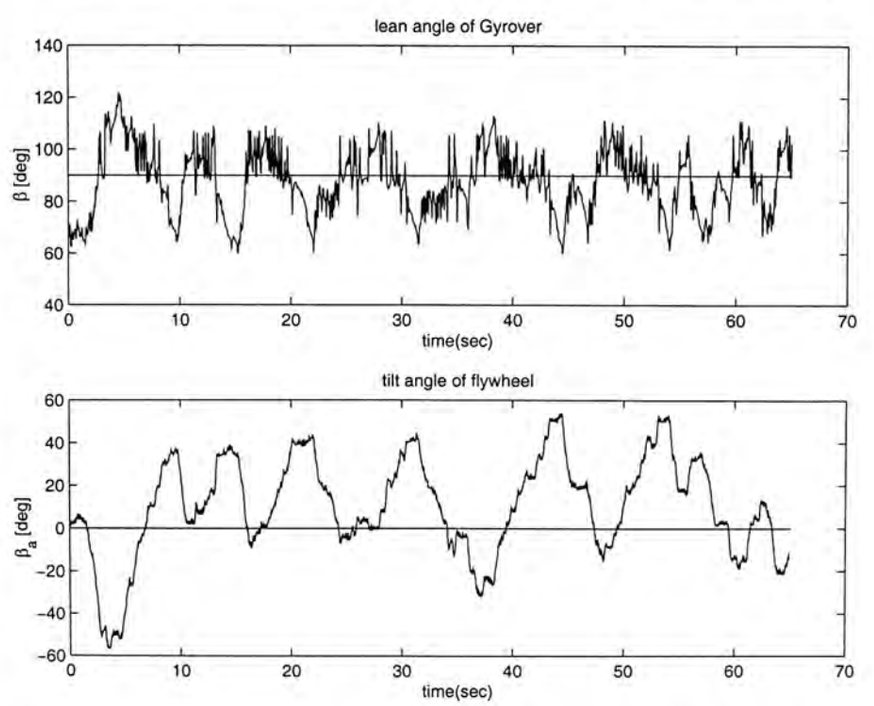


Figure 5.15: Vertical balancing by CNN model, trail #3.

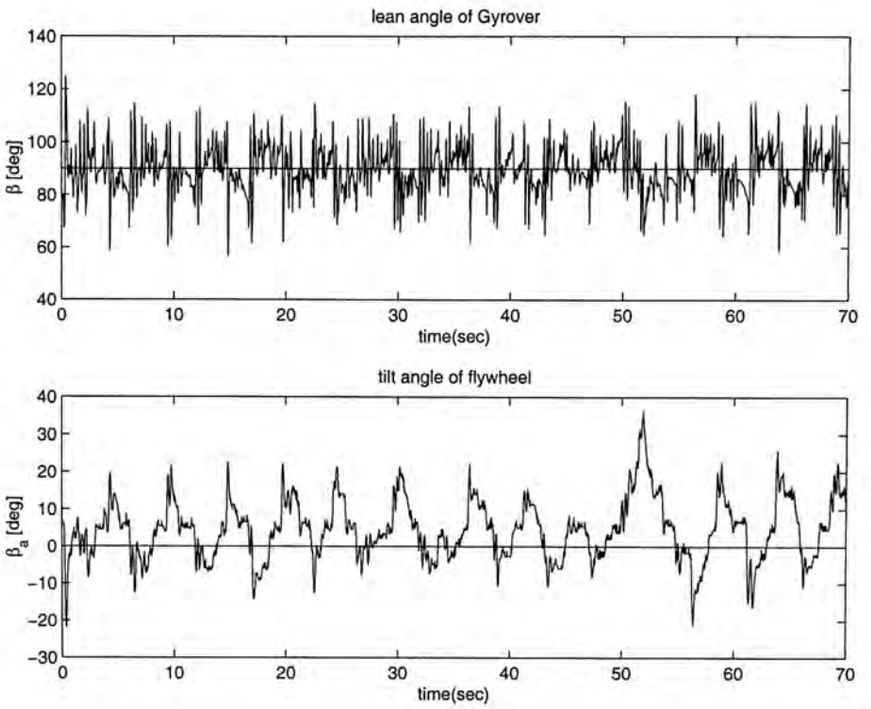


Figure 5.16: Vertical balancing by human operator.

5.2.2 Tilt-up motion

Next, we implement another CNN model which is trained by human tiltup motion data, the results are shown in Figure 5.17 and 5.18 for CNN model control, while the human control is shown in Figure 5.19. The performance of these motions are summarized in Table 5.4.

	average lean angle $\bar{\beta}$	$DOF_{flywheel}$
CNN control #1	97.26^o	0.6774
CNN control #2	95.60^o	0.4039
Human control	87.31^o	0.7372

Table 5.4: Performance measures for tiltup motion.

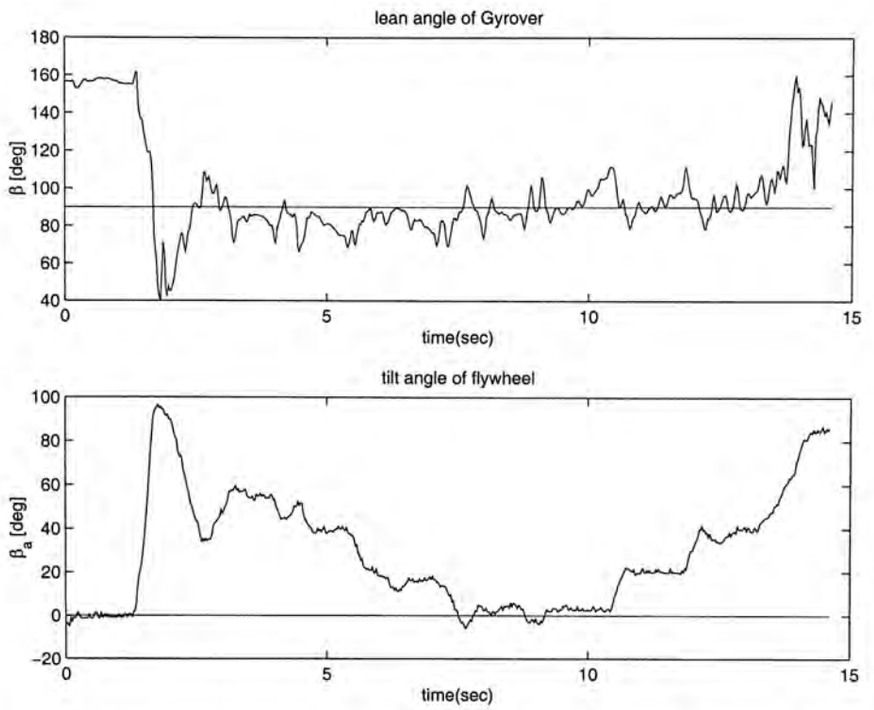


Figure 5.17: Tiltup motion by CNN model, trail #1.

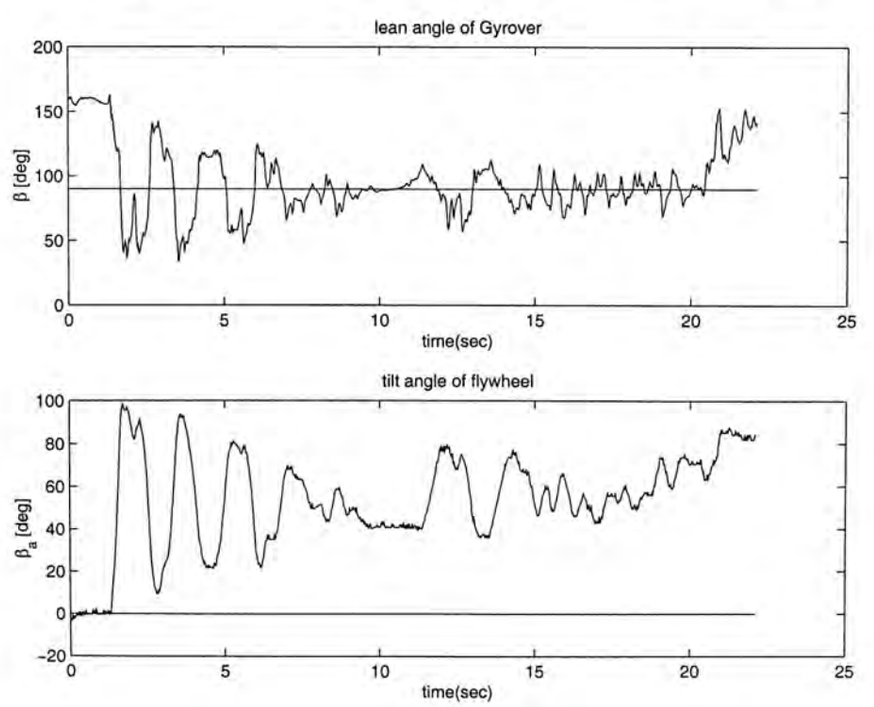


Figure 5.18: Tiltup motion by CNN model, trail #2.

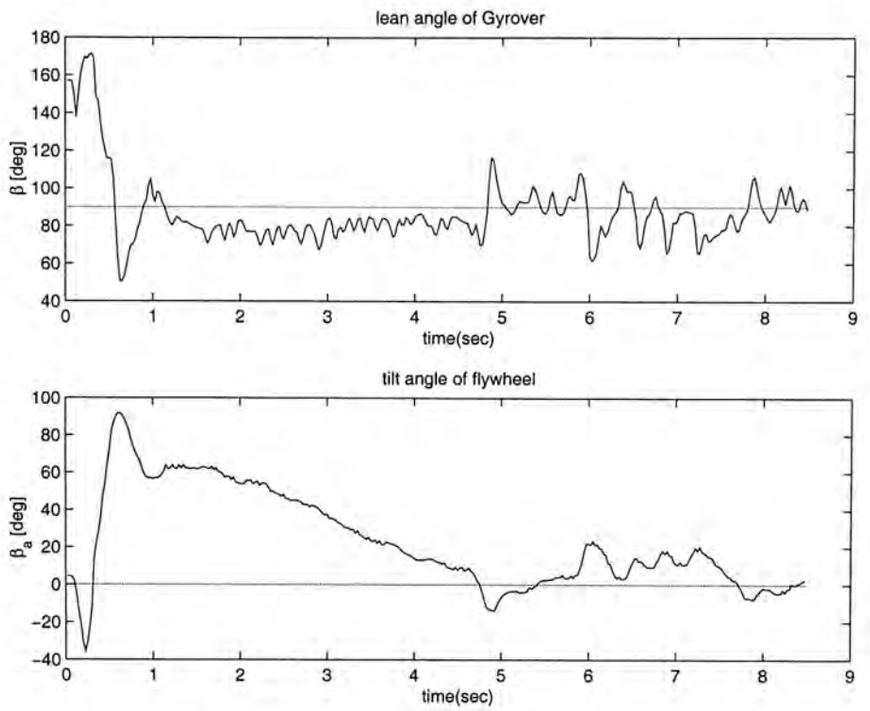


Figure 5.19: Tiltup motion by human operator.

Since a large portion of the flywheel's motion is contributed to tiltup the robot, the overall degree of freedom of the flywheel in tiltup motion is much lower than that of lateral stabilzation. For the CNN model control in Figure 5.17 and 5.18, the robot is lying on the ground initially, with $\beta \approx 150^o$, after a few seconds, the model tiltup the robot and brings the robot back to the upright position.

5.3 Combined motion

We observed that the CNN models for lateral balancing and tiltup motion are subjected to some intial condition, the problem can be solved by combining the two motions to form a single motion.

Consider the case that the robot is in the fall position, that is, with $\beta \approx 150$. In Figure 5.17 and 5.18, although the CNN tiltup model is able to keep the robot

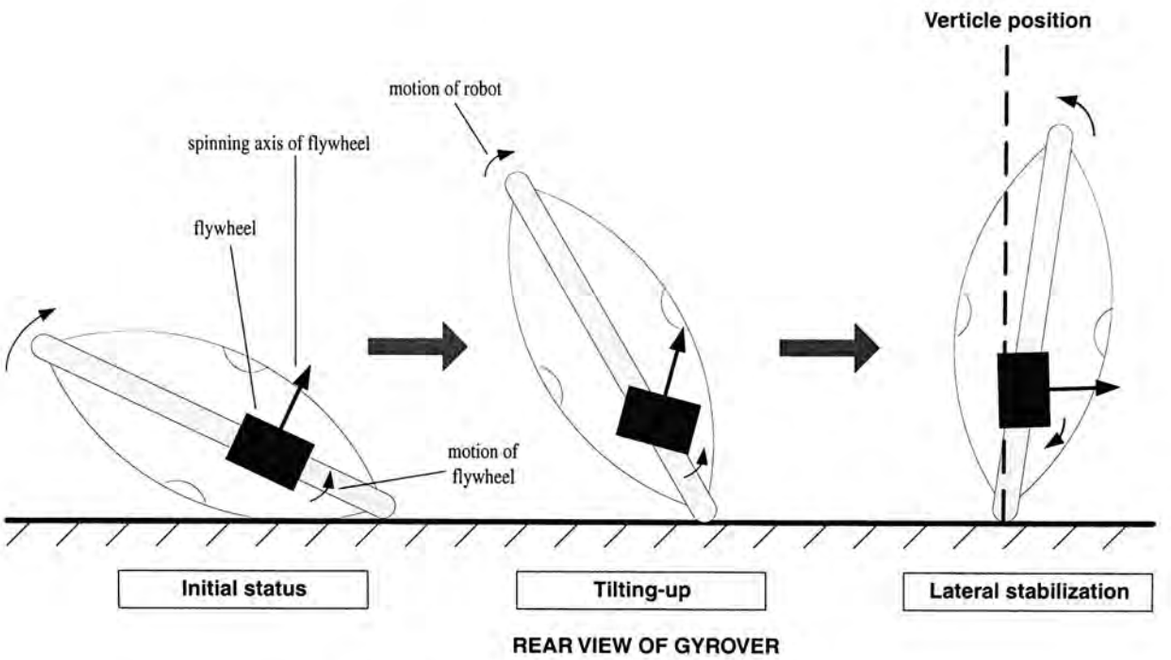


Figure 5.20: Combined motion.

to stay around at 90^o for a certain moment, the robot will fall back to the ground eventually because the flywheel has reached an ill-condition ($\beta_a = \pm 90^o$). Moreover, the tiltup model is unable to let the robot to converge to 90^o sometimes, which causes a large fluctuation in the lean angle about 90^o, Figure 5.21.

To deal with this problem, we combine the tiltup motion together with the lateral balanced motion, Figure 5.20. Since the CNN model is unable to keep the robot at the vertical position, after the robot has tiltup, we ask the model to balance the robot at 90^o.

The experimental result for the whole tiltup and stabilzation process after the combination is shown in Table 5.5 and Figure 5.22. Initially, the robot is in a fall position, by executing the tiltup control of the CNN model, the robot is recovered to the vertical position. Afterwards, the lateral stabilization is controlled by another model which specifically trained for keeping the robot into the vertical position.

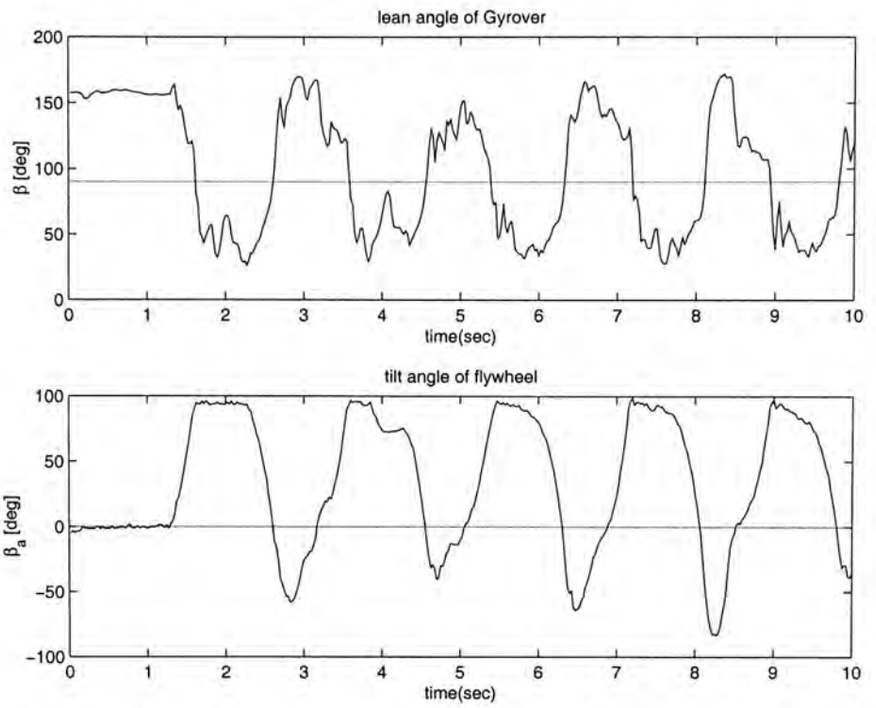


Figure 5.21: Fluctuation in the lean angle made by the tiltup model.

From the results, the combined motion can keep the robot at the vertical position well after tiltup from the ground for a much more longer period of time.

	average lean angle $\bar{\beta}$	$DOF_{flywheel}$
CNN control #1	88.40^o	0.8998

Table 5.5: Performance measures for combined motion.

5.4 Discussions

In this chapter, the CNN models for lateral balancing and tiltup motion are being verified by experimental implementations. By combining the two motions into a single motion, the robot is able to recover from the fall position, and then to remain stable at the vertical position after tiltup. Therefore, we have completed the low-level behavior module within the behavior-based architecture shown in Figure 4.5.

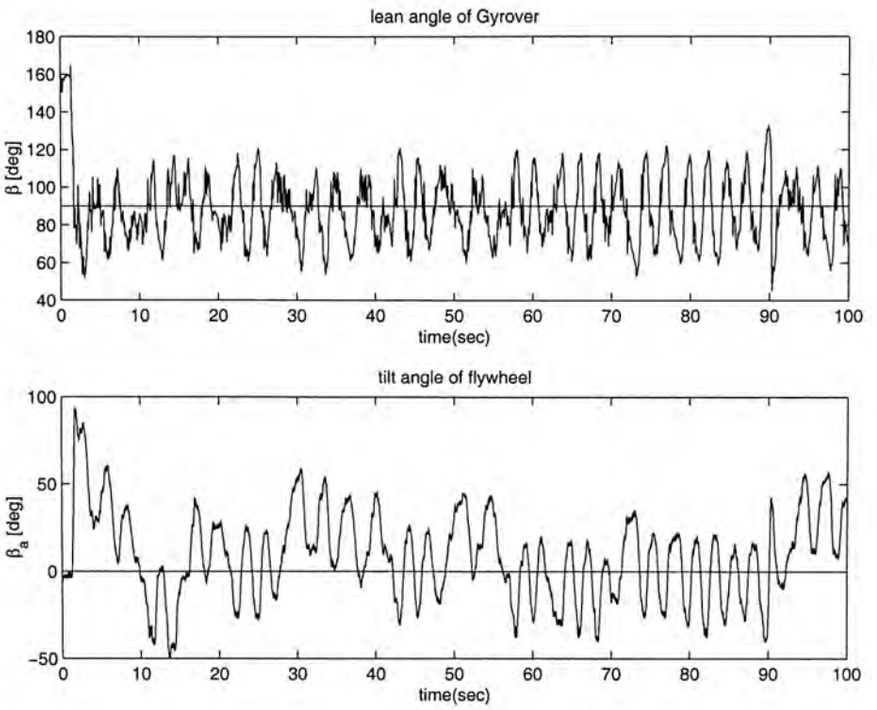


Figure 5.22: Tiltup and vertical balanced motion by CNN models.

With this module completed, we are going to develop a semi-autonomous control for Gyrover in the next chapter.

Chapter 6

Shared Control

Based on the successful implementations of the lateral balancing and tiltup motion, in this chapter, we are going to develop a shared control framework for Gyrover. In fact, any situation of a system using shared control will involve human interactions. Under shared control, the human operator acts as a supervisor for the overall control, while the robot itself can handle some local motions which in turn to assist the human in control. In order to distribute the control tasks systematically, we develop an expression to make such a decision. Experimental results will be given in order to verify our idea.

6.1 Concept

In fact, shared control happens in many daily examples, especially for human-animal interactions. First of all, let's consider the horse riding case [32], it is a fairly good example of semi-autonomous systems, or more specifically, shared control system.

For the horse which is being riden by human, it is usually able to take care of all low-level tasks such as coordination of leg motions, stability, local obstacles avoidance and provide enough power and speed for different actions. On the other hand,

the rider provides global planning, interacts with the horse to arrive at different locations and achieve various goals. At the same time, the rider can override any horse behavior by pulling the reins or hitting on the horse's body if necessary. Throughout the journey, the rider relies on the horse motoric abilities and the horse's behaviors become more intelligent by getting the rider's command. The interaction between the two individuals happens in a natural and simple way.

Another example we want to illustrate is to ask a robotic arm to handle a cup of tea [39]. The whole task can be decomposed into two subtasks: (i) to handle the cup of tea safely without pour the tea (local balancing), and (ii) to reach the desired location (global navigation). In a teleoperated environment, it maybe difficult for a human operator to perform both tasks simultaneously, or it would be mentally taxing. However, if an autonomous module is introduced for the local stability of the cup, the operator in the control loop only responsible for the navigation task, which greatly reduce the burden for the operator. Moreover, it is clear that the performance of the system would be much better and stable than being controlled by a single entity (human/machine).

Gyrover is a complex system not only in terms of the difficulties in deriving its mathematical model, but also in terms of its control by human operator. The robot can be controlled manually through a radio transmitter with two independent joysticks, one of them is assigned to control the drive motor, while the other one is assigned to control the tilt motor. Similar to a bicycle, Gyrover is a single track vehicle which is inherently unstable in its lateral direction. Therefore, different from controlling a quasi-static mobile robot, the human operator not only handles the global navigation for the robot, but also needs to pay attention to govern the lean angle of the robot simutantously. Moreover, the highly coupling effect between the

wheel and the internal flywheel also complicates the control of Gyrover. To this end, for such a complex system, instead of making a fully autonomous control, it is much more practical to develop a control method which can "share" the workload of human operator.

Recently, shared control has been widely applied into many robotics man-machine systems, from health care [31, 32, 37, 40, 41, 43] to telerobotics [33, 34, 35, 39, 42]. For rehabilitation applications, a typical example is robotic wheelchairs. Although the wheelchair itself can provide a level of autonomy for the users, it is still desirable that the user can augment the control by the on-board joystick in some special occasion (e.g. docking, pass thru a doorway). A telerobotic system usually consists of a human operator and several autonomous controllers. Human operator usually interacts with the system in different ways. One of the important issues is to develop an efficient method to combined human and machine intelligences so that the telerobotic system can perform tasks which cannot be done by either human or autonomous controller alone [35]. In these shared control system, the autonomous modules exist in the system assist the human operator during navigations, in order to relief the tensions of the operators in a complex system. Usually, the human operator is responsible for some high-level control (e.g. gobal navigation), while the machine performs low-level control (e.g. local obstacles avoidance).

In fact, the two behaviors we have mentioned in the previous chapters, (i) Lateral balancing and (ii) Tiltup motion, are designed to tackle the robot's instability problem in the lateral direction. Since we have successfully modeled and implemented the two behaviors by a machine learning approach and verified in experiments, the next step is to incorporate these motions with human control in order to develop a shared control framework for Gyrover. We prefer using a shared control scheme

rather than a fully autonomous one because of the following reasons:

- **Sophisticated dynamic system.** As mentioned before, it is difficult for us to obtain a complete mathematical model to govern the motions of Gyrover, due to its complicated dynamic and nonholonomic nature. This makes us encounter many difficulties in developing a fully autonomous system for Gyrover at this stage.

- **Hardware limitations.** Due to the special physical structure of Gyrover, the current prototype of Gyrover we are using still does not have any navigation devices equipped on-board (e.g. vision), which is impossible for the robot to navigate itself.

- **Importance of human operators.** Practically, for some complicated tasks, which may be trival for humans, robots often do not perform well. Therefore, human operator is essential to exist in the control loop in order to monitor and operate the executive system.

- **Time and cost.** Building a fully autonomous system which provides safe and robust performance would be time consuming and costly, in terms of computations and resources. In contrast, it is far more practical and much cheaper to develop a semi-autonomous system.

- **Accuracy vs Reliablilty.** Machines are excellent in performing repetitive tasks quickly and accurately but their abilities to adapt changes in environment is low. On the other hand, humans are usually reliable, with tremendous perception ability and good decision making in unpredictable situations, but

their accuracies are relatively lower than machines. Shared control can let them compensate each weakness which would result a better control.

- **Teleoperations.** Gyrover can be operated by humans through a radio transmitter, which allows humans to participate in the control of the robot.

The main difficulty in developing a shared control for Gyrover is due to the access of the tilt motor. Since the lean angle of the robot is controlled by the tilt motor, not only the autonomous module will access the tilt motor to achieve stability in the lateral direction, the human operator also need to access the tilt motor during navigation. At a particular time instant, these commands may contradict with each other. Therefore, it is a big issue to let the system to decide which command is going to be executed, and at the same time, to manage the contaminated commands with a reasonable way. To this end, we have developed an expression for making this decision, which will be discussed in the later part of this chapter. With a better sharing between the machine and human operator, the performance of the system can be enhanced, and the range of tasks that can be performed by the system can also increase.

6.2 Schemes

In fact, there are many aspects of "sharing" in shared control, varies from application to application. Basically, a semi-autonomous control can be categorized into serial type and parallel type [39]. In serial type, the manual control and autonomous control cannot be executed simultaneously, only one of them will be selected at a time; in parallel type, both manual and autonomous control can be executed simultaneously.

In the following sections, we will breifly discuss three operating modes of shared control, namely: (1) Switch mode, (2) Distributed mode, and (3) Combined mode.

6.2.1 Switch mode

In switch mode, the manual control and autonomous control are switched in serial, as shown in Figure 6.1. The condition to trigger the switch depends on applications, for example, if an operator is acted as a supervisor of the control system, the human control will only be activated whenever the system reaches an "ill condition". No matter which control module is switched, the robot will be fully controlled by the selected one. If a high cooperation between the machine and operated is required, we must have a function (Π) which can "smartly" switch between the two control module.

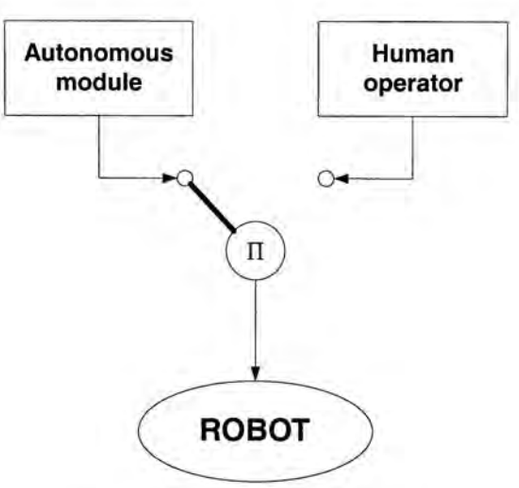


Figure 6.1: Switch mode.

6.2.2 Distributed mode

Figure 6.2 illustrates the architecture of distributed control. Different from switch mode, both manual and autonomous control can be executed in parallel in this

mode. The control of various actuators (u_i) in the entire system will be distributed to either of the two modules.

Therefore, the two entities can exist in the system peacefully without disturbing each other. However, this also shows the weakness of this mode because there is no communication bewteen the two entities. The operator cannot modify the commands from autonomous module even the robot is performing or tends to perform some undesirable motions.

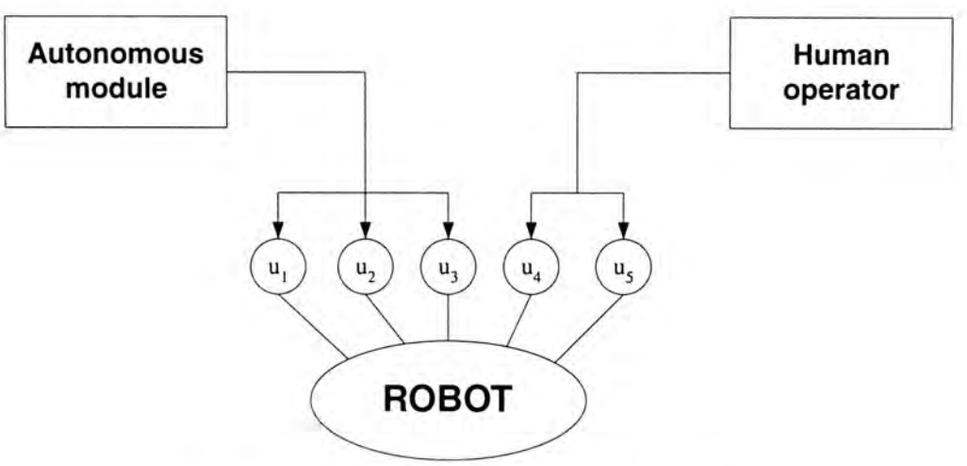


Figure 6.2: Distibuted control mode.

6.2.3 Combined mode

Combined mode is in fact an extension of distributed mode, Figure 6.3. However, the input to a single actuator is a combination of the operator's command and the machine command. There are many ways to combine the output vectors from the task modules: a simple summation, a simple average, weighted sum and average, voting on angle and velocity, and some unusual variations. In practice, the weighted average performs well since it is not computationally expensive and its performance is predictable [42].

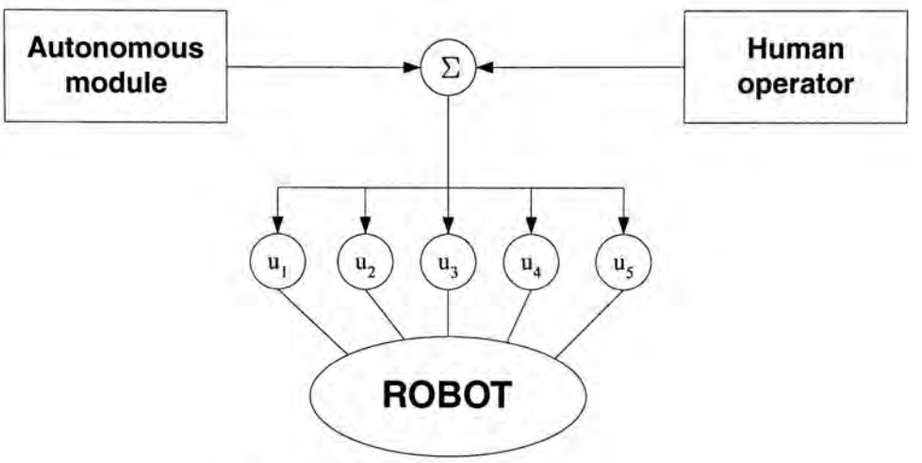


Figure 6.3: Combined mode.

6.3 Shared control of Gyrover

Analog to the example of handling a cup of tea, in our approach, in order to reduce the operator burden in controlling a statically unstable robot, it is desired that Gyrover itself can maintain a degree of local balancing, while the operator only responsible for the navigation task. In considering which mode of sharing is suitable for Gyrover shared control, we found that the commands from the automation module (lateral balancing and tiltup) always contradicts with the navigation commands. It is due to the special steering mechanism of Gyrover, which is entirely contributed by the tilting effect of the internal flyhwheel.

As mentioned in section 3.2.1, when a disc is rolling, it will steer to the direction that it is leaning. Since the autonomous module is designed to keep the lean angle into the vertical position, if we attempt to steer to the left/right manually (i.e. lean to left/right), the machine will generate commands to stabilize the robot back to the vertical position, which will totally oppose the changes we want to make. Therefore, the commands from the two modules is impossible to combine into a single valid command during navigation. Fortunately, this problem is solved automatically if we

consult the behavior-based control architecture we disccussed in Chapter 2.

Referring to Figure 4.5, the mid- and high-level behaviors are replaced by human operator in shared control. Due to the high flexibility of the subsumption architecure, we obtain the shared control architecture as shown in Figure 6.4, without destorying the original control structure, which shows the beauty of behavior-based control architecture. Since the navigation tasks are entirely given to the human operator, the operator will solely control the drive motor throught a radio transmitter. On the other hand, we suppose the robot can maintain lateral stablility when it stops rolling, or when a complete fall is detected, it will automatically tiltup back to its upright position. Thus, the tilt motor is jointly control by the operator and the machine.

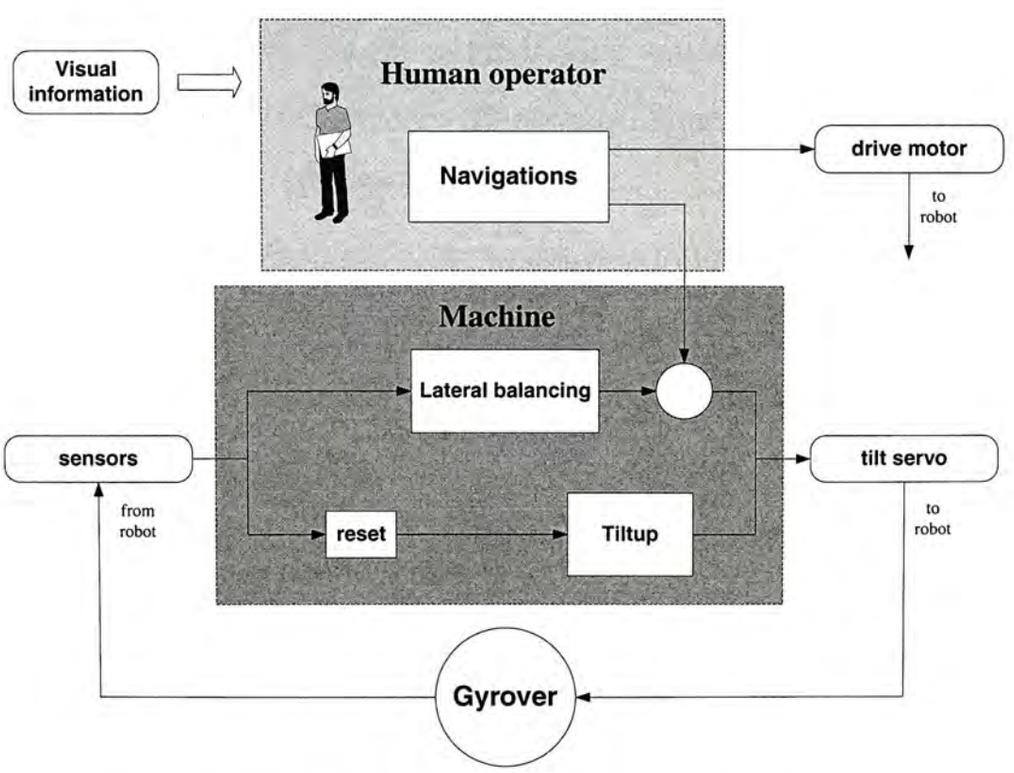


Figure 6.4: Subsumption architecture of shared control.

According to Figure 6.4, regarding to the tilt motor, switch mode is used since

the operator and the machine cannot control the motor at the same time; regarding to the whole structure, the system is somewhat in a distributed mode of sharing. As a result, the shared control of Gyrover combines the switch mode and the distributed mode, which compensates each mode's weakness.

6.4 How to share

Recalling the horse riding example, it is believed that the horse acknowledges the rider commands if they exceed a certain threshold. This threshold may depend on the horse training (reliability of the autonomous system), the skill of rider, and on the situation at hand. If the rider wishes to correct or modify the horse current behavior, he/she will increase the level of stimulus which is acted on the horse (pulling the reins more or pushing harder on the saddle). This continues until the horse changes its behavior as wished by the rider. A poor communication or compromisation between them can lead to undesirable or even dangerous results. Therefore, in this section, we develop a function to decide whether to follow or neglect the commands from the online operator.

First of all, let's introduce the variables that constitute the function, which are similar to those proposed in [32]:

1. **Degree of Autonomy**, A where $0 \leq A \leq 1$.

   This is a parameter which can be adjusted by the online operator. If the operator (a novice) wish to rely much more on the autonomous module, he/she should select a higher value of A at the beginning of an operation. Otherwise, if an experienced operator is confident with his/her control skill, a lower value of A can be selected. We will demonstrate the effect of this parameter later.

2. **Strength of conflict**, S where $0 \leq S \leq 1$.

   This parameter measures the conflict between the operator and the current status of the system, it will vary from time to time whenever the operator is given a command to alter the system's trajectory. A high value of S indicates that the operator is making a control command which greatly affect the current status of the system, while a low value of S indicates that only a small disturbance is generated. This value will pass to the function instanteously to make a decision whether to execute the operator's command or not. The strength of conflict S can be defined as:

$$S_\beta = \frac{\partial \beta}{\partial \beta_{\max}} \qquad \text{or} \qquad S_{\text{out}} = \frac{|u_{\text{operator}} - u_{\text{machine}}|}{\partial u_{\max}} \tag{6.1}$$

   where S_β is measured in terms of the changes in the lean angle β of the robot, S_{out} is in terms of the conflict between the command from operator and the machine.

3. **Confidence level**, C where $0 \leq C \leq 1$.

   Contradict to the strength of conflict, C is a parameter to show the confidence of an operator in making the current control command. It is obvious that the higher value of C, the more confident the operator is. This is also a time varying parameter which will pass to the function to let the system to make a decision. The confidence level C can be defined as:

$$C = \frac{|\partial u_{\text{operator}}|}{\partial u_{\max}} \tag{6.2}$$

Based on the above definition, at a particular time instant, the system recieves a command from the operator and the machine simultaneously, we obtain the following relationship between S and C:

$$\begin{aligned} \text{if} \quad & C > S, \quad \text{follow operator's command,} \\ & C \leq S, \quad \text{follow machine's command.} \end{aligned} \tag{6.3}$$

The above expressions imply that if the operator is confident enough to modify the current system trajectory, his/her command will be executed. On the other hand, if the system determines that the command of the operator is potentially to let the robot falls down, his/her command will be neglected, and the system will execute the balancing command from the autonomous module. However, the threshold of the above expressions remains constant and it is dependent on the system parameters. Practically, a system may be potentially operated by different operators, it is desired that the thershold of the decision to be dependent on the operator. To this end, we introduce the parameter of Degree of Autonomy (A) into the above expressions,

$$\begin{aligned} \text{if} \quad & C \cdot (1 - A) > S \cdot A, \quad \text{follow operator's command,} \\ & C \cdot (1 - A) \leq S \cdot A, \quad \text{follow machine's command.} \end{aligned} \tag{6.4}$$

By rewriting equation (6.4), we have,

$$\Pi(A, S, C) = \lambda \cdot C - S \tag{6.5}$$

where $\lambda = (1 - A)/A$ for simplicity, and the decision finally becomes,

$$\begin{aligned} \text{if} \quad & \Pi(A, S, C) > 0, \quad \text{follow operator's command,} \\ & \Pi(A, S, C) \leq 0, \quad \text{follow machine's command.} \end{aligned} \tag{6.6}$$

The function Π is called a decision function which allows a system to decide whether to execute the command from operator in a shared control environment. To validate the decision function, we let $A = 0$, which implies that the operator do not need any assistance from the autonomous module and the system should respond to all the commands from the operator. From equation (5.6),

$$\Pi(0, S, C) = +\infty > 0 \qquad \forall S, C$$

$\Pi(0, S, C)$ is always positive so that the system always execute the commands from the operator. Now, consider when $A = 1$,

$$\Pi(1, S, C) = -S \leq 0 \qquad \forall S, C$$

$\Pi(1, S, C)$ always be negative or equal to zero, which implies the system will totally follow the machine commands and disregard all the operator's control.

To further validate the decision function Π in (6.5), we perform the following experiments to see how the system works with this function. Table 6.1, 6.2 and 6.3 show the values obtained from the decision function Π by using $A = 0.25$, $A = 0.50$ and $A = 0.75$ respectively. Based on the decision criteria in (6.6), if $\Pi(A, S, C)$ is greater than zero, the system will execute the operator's command at that particular moment, otherwise, machine's command will be executed. In each table, a shaded value represents the system has chosen the operator's command.

When $A = 0.25$, the system will more likely to rely on the operator's control. In Table 6.1, most of the operator's commands are chosen even when the Confidence level of the his/her control is quite low (smaller ∂u). On the other hand, for a higher value of A (Table 6.3), the system relies on the machine's commands more so that the frequency of accepting the operator's commands reduces significantly.

	Current lean angle of the robot β										
∂u	40^o	50^o	60^o	70^o	80^o	90^o	100^o	110^o	120^o	130^o	140^o
0	-0.56	-0.44	-0.33	-0.22	-0.11	0	-0.11	-0.22	-0.33	-0.44	-0.55
2	-0.36	-0.25	-0.14	-0.03	0.08	0.11	-0.01	-0.12	-0.23	-0.34	-0.45
4	-0.17	-0.06	0.06	0.17	0.28	0.21	0.10	-0.01	-0.12	-0.23	-0.34
6	0.03	0.14	0.25	0.36	0.43	0.32	0.21	0.09	-0.02	-0.13	-0.24
8	0.22	0.33	0.44	0.56	0.53	0.42	0.31	0.20	0.09	-0.02	-0.13
10	0.42	0.53	0.64	0.75	0.64	0.53	0.42	0.31	0.19	0.08	-0.03
15	0.09	1.01	1.13	1.01	0.90	0.79	0.68	0.57	0.46	0.35	0.24
20	1.39	1.50	1.39	1.28	1.17	1.06	0.94	0.83	0.72	0.61	0.50
30	2.14	2.03	1.92	1.81	1.69	1.58	1.47	1.36	1.25	1.14	1.03
40	2.67	2.56	2.44	2.33	2.22	2.11	2.00	1.89	1.78	1.67	1.56

Table 6.1: Decision making of $A = 0.25$.

	Current lean angle of the robot β										
∂u	40^o	50^o	60^o	70^o	80^o	90^o	100^o	110^o	120^o	130^o	140^o
0	-0.56	-0.44	-0.33	-0.22	-0.11	0	-0.11	-0.22	-0.33	-0.44	-0.55
2	-0.46	-0.35	-0.24	-0.13	-0.02	0.01	-0.11	-0.22	-0.33	-0.44	-0.55
4	-0.37	-0.26	-0.14	-0.03	0.08	0.01	-0.01	-0.21	-0.32	-0.43	-0.54
6	-0.27	-0.16	-0.05	0.06	0.13	0.02	-0.09	-0.21	-0.32	-0.43	-0.54
8	-0.18	-0.07	0.04	0.16	0.13	0.02	-0.09	-0.20	-0.31	-0.42	-0.53
10	-0.08	0.03	0.14	0.25	0.14	0.03	-0.08	-0.19	-0.31	-0.42	-0.53
15	0.15	0.26	0.38	0.26	0.15	0.04	-0.07	-0.18	-0.29	-0.40	-0.51
20	0.39	0.50	0.39	0.28	0.17	0.06	-0.06	-0.17	-0.28	-0.39	-0.50
30	0.64	0.53	0.42	0.31	0.19	0.08	-0.03	-0.14	-0.25	-0.36	-0.47
40	0.67	0.56	0.44	0.33	0.22	0.11	0	-0.11	-0.22	-0.33	-0.44

Table 6.2: Decision making of $A = 0.50$.

The above experiments simply illustrate that the decision function Π can judge whether to execute human operator's commands effectively by taking the value A into accounts, which is very important in a shared control system.

In fact, the system neglects the operator's commands only when the command is potentially dangerous to the robot. Since a positive change in the tilt command

	Current lean angle of the robot β										
∂u	40^o	50^o	60^o	70^o	80^o	90^o	100^o	110^o	120^o	130^o	140^o
0	-0.55	-0.44	-0.33	-0.22	-0.11	0	-0.11	-0.22	-0.33	-0.44	-0.55
2	-0.49	-0.38	-0.27	-0.16	-0.05	-0.03	-0.14	-0.25	-0.36	-0.47	-0.58
4	-0.43	-0.32	-0.21	-0.10	0.01	-0.06	-0.17	-0.28	-0.39	-0.50	-0.61
6	-0.37	-0.26	-0.15	-0.04	0.03	-0.08	-0.19	-0.31	-0.42	-0.53	-0.64
8	-0.31	-0.20	-0.09	0.02	0	-0.11	-0.22	-0.33	-0.44	-0.56	-0.67
10	-0.25	-0.14	-0.03	0.08	-0.03	-0.14	-0.25	-0.36	-0.47	-0.58	-0.69
15	-0.10	0.01	0.13	0.01	-0.10	-0.21	-0.32	-0.43	-0.54	-0.65	-0.76
20	0.06	0.17	0.06	-0.06	-0.17	-0.28	-0.39	-0.50	-0.61	-0.72	-0.83
30	0.14	0.03	-0.08	-0.19	-0.31	-0.42	-0.53	-0.64	-0.75	-0.86	-0.97
40	0	-0.11	-0.22	-0.33	-0.44	-0.56	-0.67	-0.78	-0.89	-1.00	-1.11

Table 6.3: Decision making of $A = 0.75$.

will give a positive change in the lean angle of the robot, if the lean angle is beyond 90^o, a larger ∂u will make the lean angle grows bigger, which potentially to make the robot falls down. Therefore, in this case, if the operator is not confident enough to make this change, his/her command will be neglected.

6.5 Experimental study

In this section, we implement the shared control framework as shown in Figure 6.4, by applying the decision function we have mentioned in the last section. We have designed several tasks for the robot to perform under the shared control scheme, including (i) heading control (ii) a straight path tracking, (iii) a circular path tracking, and (iv) point-to-point navigation.

Since the autonomous module now in hand is only responsible for the lateral stabilization and tiltup motion when the robot is held in a stationary location, the navigation task of the robot will be entirely given to the human operator to control, which implies that the human cannot rely on the machine throughout the

navigation. Based on this limitation, we use a relatively high level of autonomy ($A \approx 0.25$) in Gyrover shared control. From the experiments, we can observe that even the operator has shared a level of control to the system, the robot can still achieve some basic goals in mobile teleoperations.

6.5.1 Heading control

The purpose of this experiment is to illustrate the cooperation between the human operator and the autonomous module in a shared control environment. One special feature of Gyrover is the ability to turn into a desirable heading direction at a stationary location, this motion can be achieved by controlling the lean angle of the robot (left/right) until the desired heading direction is reached.

When the robot is not rolling, the system will automatically execute the lateral balancing module in order to maintain its lateral stability, by controlling the tilt motor. If the operator wishes to command the robot to turn into a particular heading angle, he/she requires to make the robot to lean at a certain angle by controlling the tilt motor also, in this case, the robot must stops the autonomous module and execute the operator's command. Therefore, if the system cannot make a right decision, the operator can never control the robot to turn into a desired heading direction.

The result of using $A = 0.2$ and $A = 0.8$ in the heading control test is shown in Figure 6.5 and Figure 6.6 respectively. For $A = 0.2$, the operator triggers the control of tilt motor at $7.5 \leq t \leq 9.5$ and $14.5 \leq t \leq 17$, in order to make the robot leans to a particular heading angle. It is clear that the operator augments the control in these periods successfully, which is expected when a low degree of autonomy is used. When there is no command from the operator, the robot will

execute the lateral balancing control from the autonomous module in order to keep the robot stays around 90^o. For $A = 0.8$, the control trajectory of the operator is completely different from the final control output to the system. The operator wants to trigger the tilt motor, but the system neglects most of his/her commands and continues to execute the lateral balancing commands from the autonomous module. The system will only execute those commands from the operator only when the particular command is greatly contribute in keeping the robot in 90^o, or when the confidence level is high, for instance, at $t \approx 13$ and $t \approx 17$.

6.5.2 Straight path

In the straight path test, the operator is asked to control the robot to travel a straight path, approximately 44 ft long. The experimental setup is shown in Figure 6.7. Three trails are given in this experiment, the trajectory that the robot has travelled in each trail is shown in Figure 6.10. The sensor data of the robot in trail #3 are plotted in Figure 6.11.

Under a shared control, although some of the control commands are being neglected by the system (flattened peaks in the final output of tilt motor command), the operator is still able to control the robot to travel a nearly straight path, with an average 0.1736 ft offset from the desired path. At $t = 9$, the robot recieved no commands from the operator and started to execute the lateral balancing module to balance the robot. As mentioned earlier, the control of the drive motor is entirely given to the operator, therefore, the system will not interfere the drive motor command, which directly follows the control of the operator.

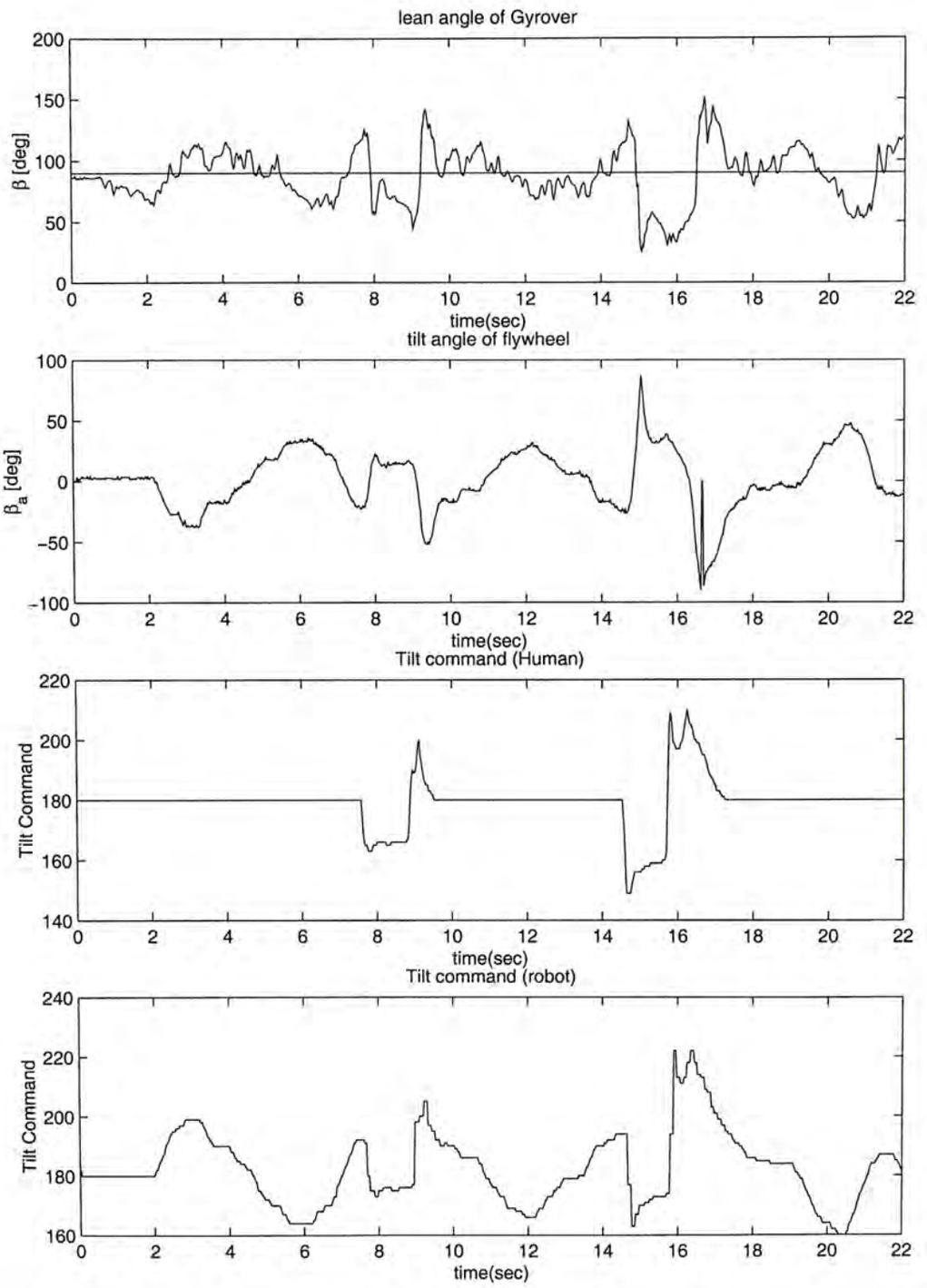


Figure 6.5: Sensor data acquired in the heading control test, $A = 0.2$.

6.5.3 Circular path

Similar to the straight path test, the experimental setup is shown in Figure 6.8. This time, the operator is required to control the robot to travel a circular path.

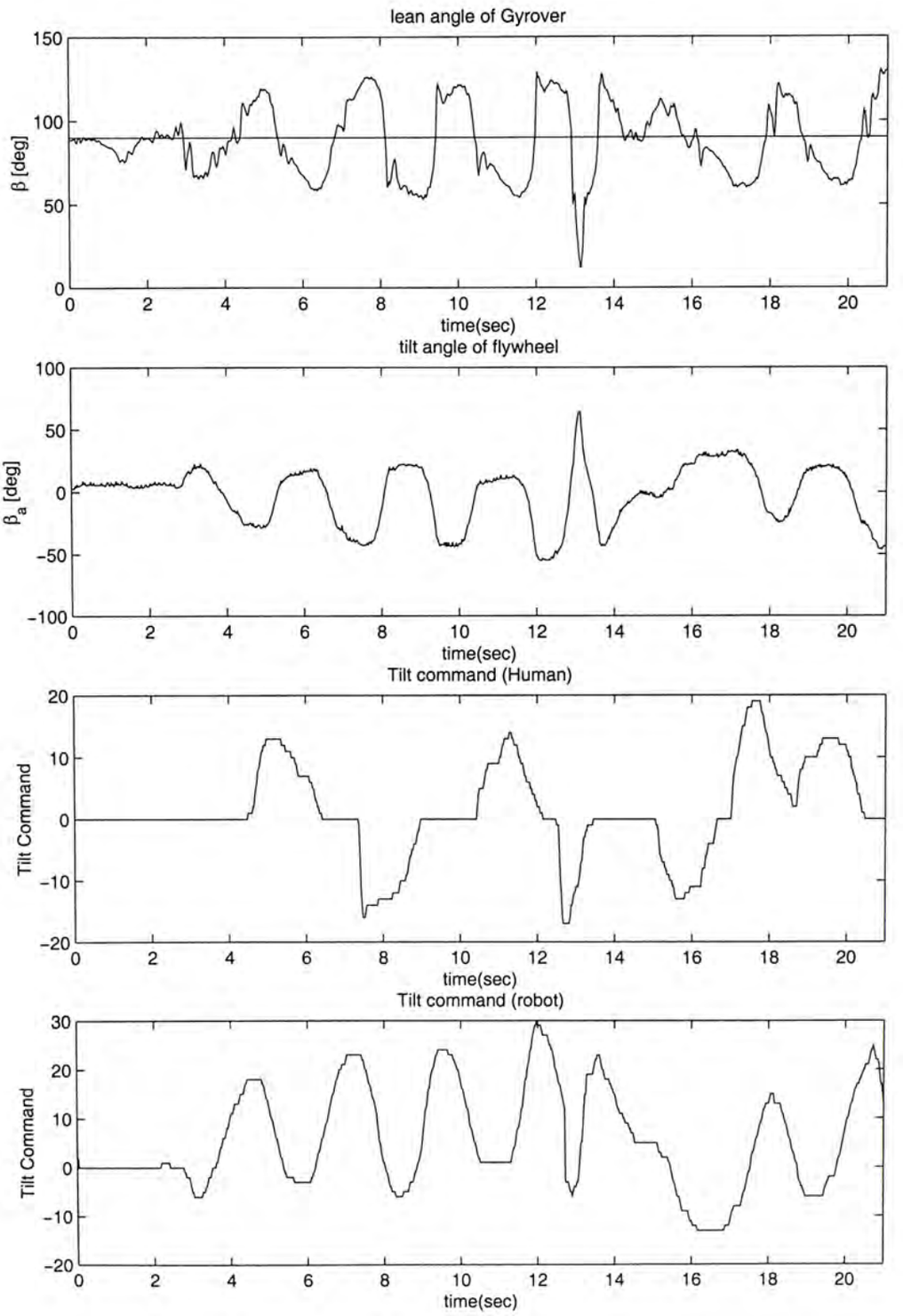


Figure 6.6: Sensor data acquired in the heading control test, $A = 0.8$.

In order to make the robot to turn in place, the operator needs to tilt the internal flywheel to make a "lean steering" precisely. If the robot fails to follow the right

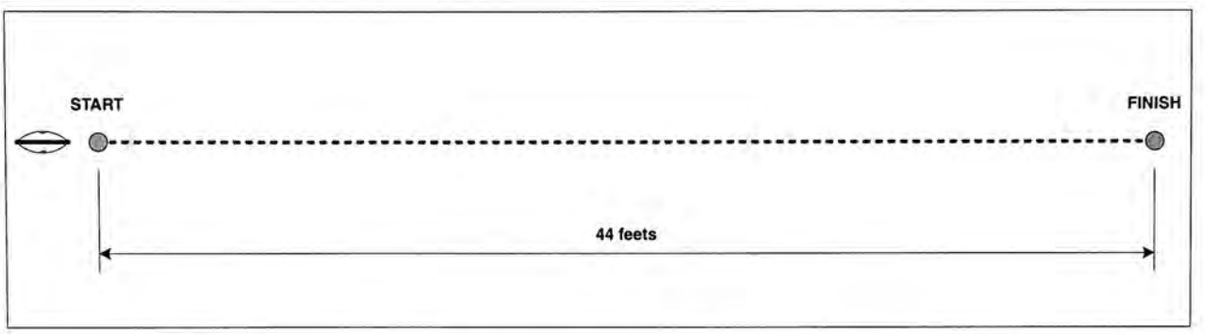


Figure 6.7: Experiment on tracking a straight path under shared control.

commands, it is unable to steer well. Figure 6.12 indicates the desired path and the actual path travelled by the robot respectively. Figure 6.13 shows the corresponding sensor data (trail #3) of the robot during travelling a circular path.

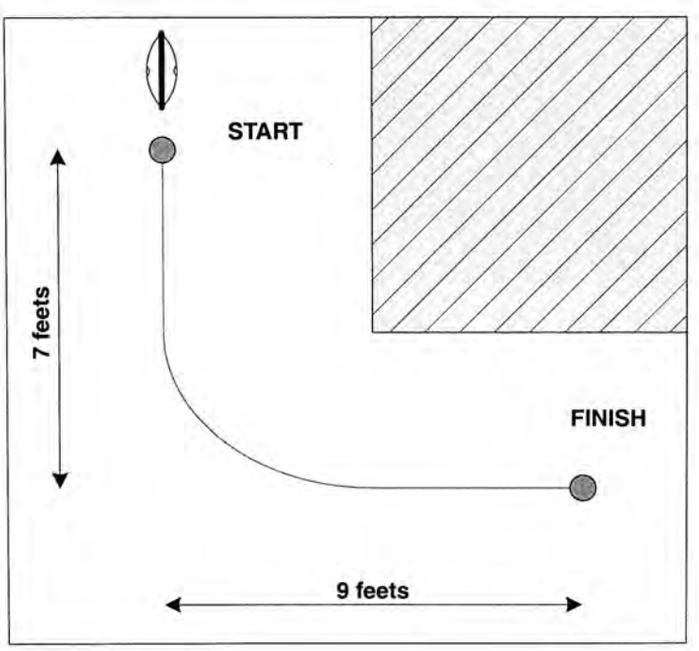


Figure 6.8: Experiment on tracking a curved path under shared control.

The average offset in the circular path test is 0.51 ft. Although the robot cannot track the circular path precisely, the operator can control the robot to move back to the goal location within 0.25 ft nearly the end of the experiments. Therefore, with a degree of shared control with the robot, the operator is still able to control the

robot to turn a tight corner.

6.5.4 Point-to-point navigation

In this experiment, we require the robot to travel from one location to anothter which are seperated by a right corner and they are far apart ($\approx$ 60 ft), Figure 6.9. The operator needs to control the robot to move from a starting area to a specific destination, which is a 2 ft $\times$ 2 ft region (the dimension of Gyrover is about 1.5 ft $\times$ 0.8 ft as viewed form the top). This experiment has two main goals:

1. The robot must reach the destination within the specific area.

2. After the robot has reached the destination, it is required that the robot can maintain its lateral balance even when the operator does not further control it.

The experimental results are shown in Figure 6.14 and 6.15.

Although we are not concerning whether the robot can accurately track the path or not, the overall offset from the path is 1.18 ft, which is an acceptable value for a 60 ft long journey. Moreover, for the three trails in this experiment, all the trajectories of the robot are converging to the destination at the end of the path. From Figure 6.15, when $t \geq 14$ (at the destination), the operator did not command the robot anymore, however, the robot can balance itself ar around 90^o. Therefore, under a shared control environment, with the human operator responsible for the navigation task of the robot, the robot is able to move from one location to another location which is far apart, and to balance itself at the vertical position when the robot stops moving (with no operator's command).

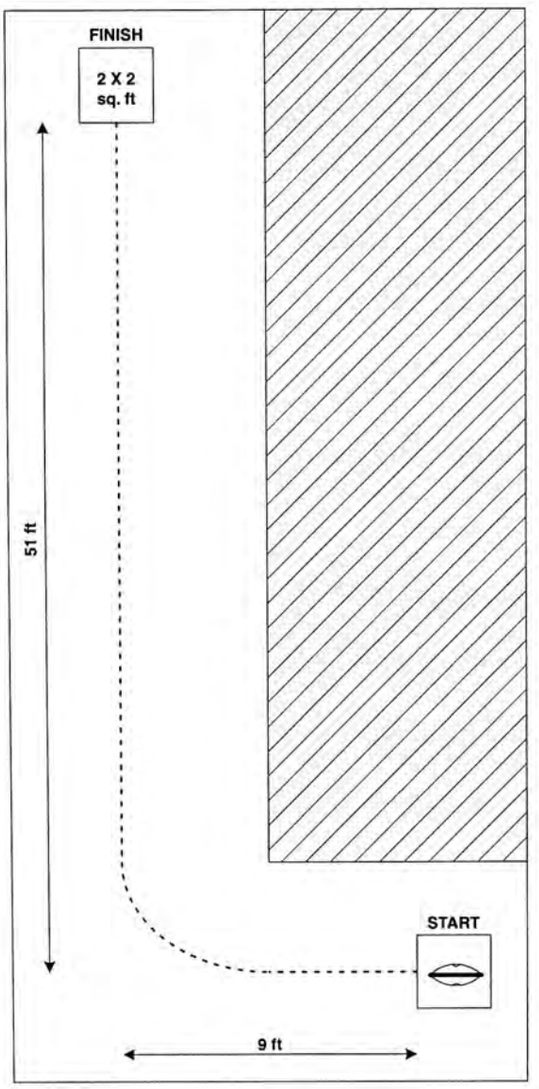


Figure 6.9: Experiment on point-to-point navigation under shared control.

6.6 Discussions

From the results we conducted from the previous experiments, we verify that our proposed shared control algorithm can let the system choose between human operator's control commands or the commands from the autonomous module systematically. Whenever the operator has chosen a high level of autonomy, the system will execute the command from the the autonomous module unless the operator has given a command which is 'confident' enough to overcome the conflict between the

operator and the machine. On the other hand, if a low degree of autnomy is chosen, the system will follow the operator's command unless a 'significant' error/conflict is measured. The proposed shared control algorithm is able to allow two entities (human and machine) to exit in the same system simultaneously.

Although Gyrover do not have an autonomous module to navigate itself to travel from one location to another, this can be done by sharing the navigation task to the operator. Under shared control, the robot will maintain its lateral balance when the operator does not command it. On the other hand, under a degree of sharing, the operator is still able to control the robot to do some specific tasks (straight path tracking, point-to-point navigation, etc). It is believed that if an autonomous navigation module exists in the system, the operator can share more naviagtion control to the machine using the proposed shared control algorithm, which can greatly reduce the duty of the online human operator.

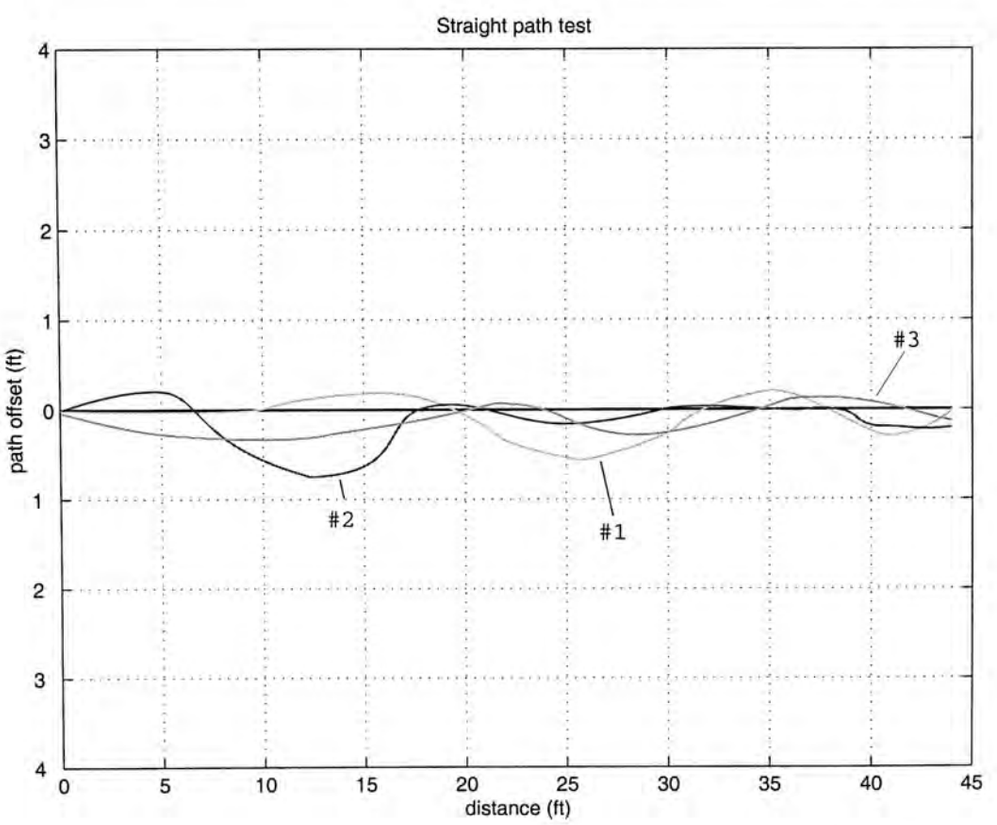


Figure 6.10: Trajectory travelled in the straight path test.

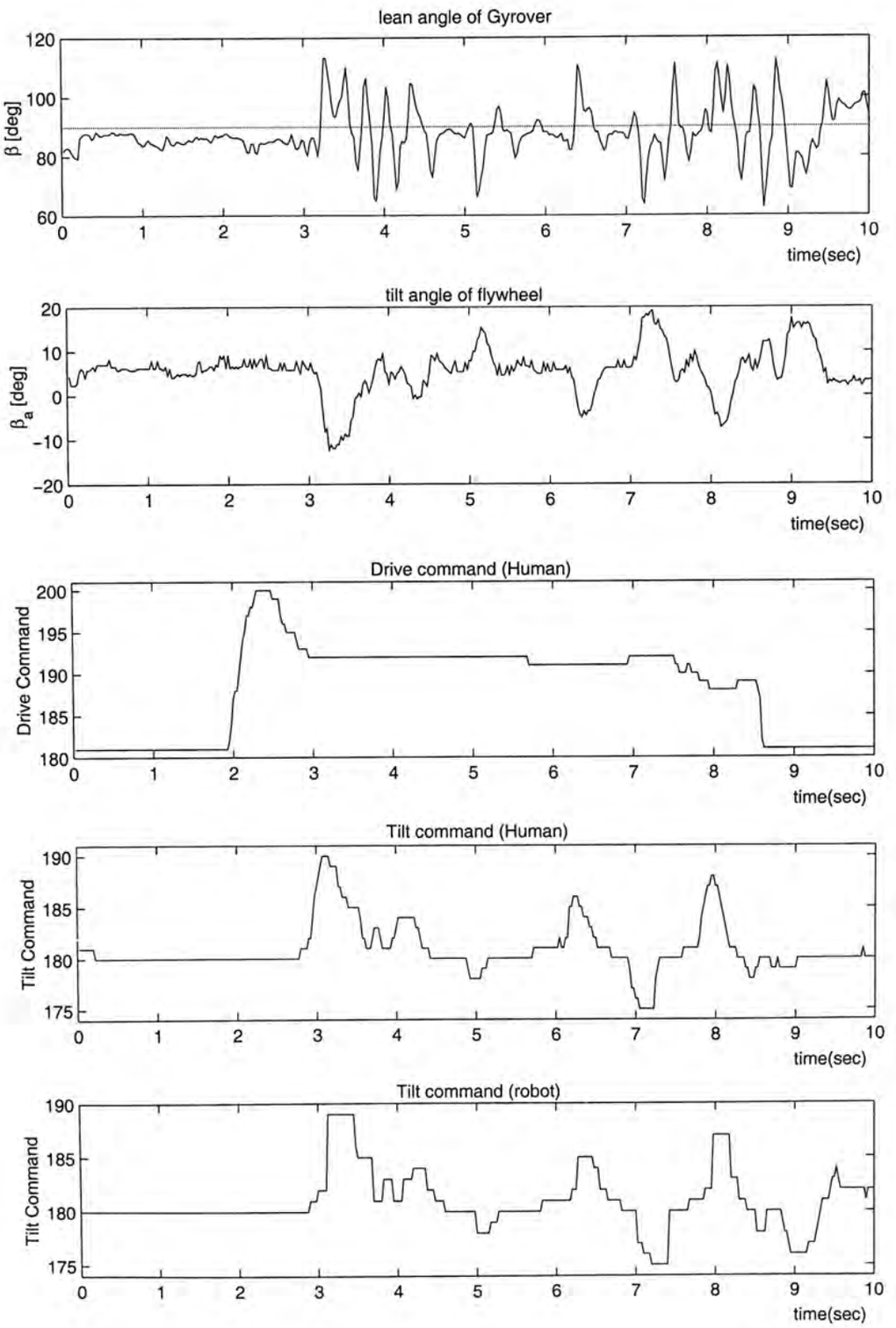


Figure 6.11: Sensor data acquired in the straight path test.

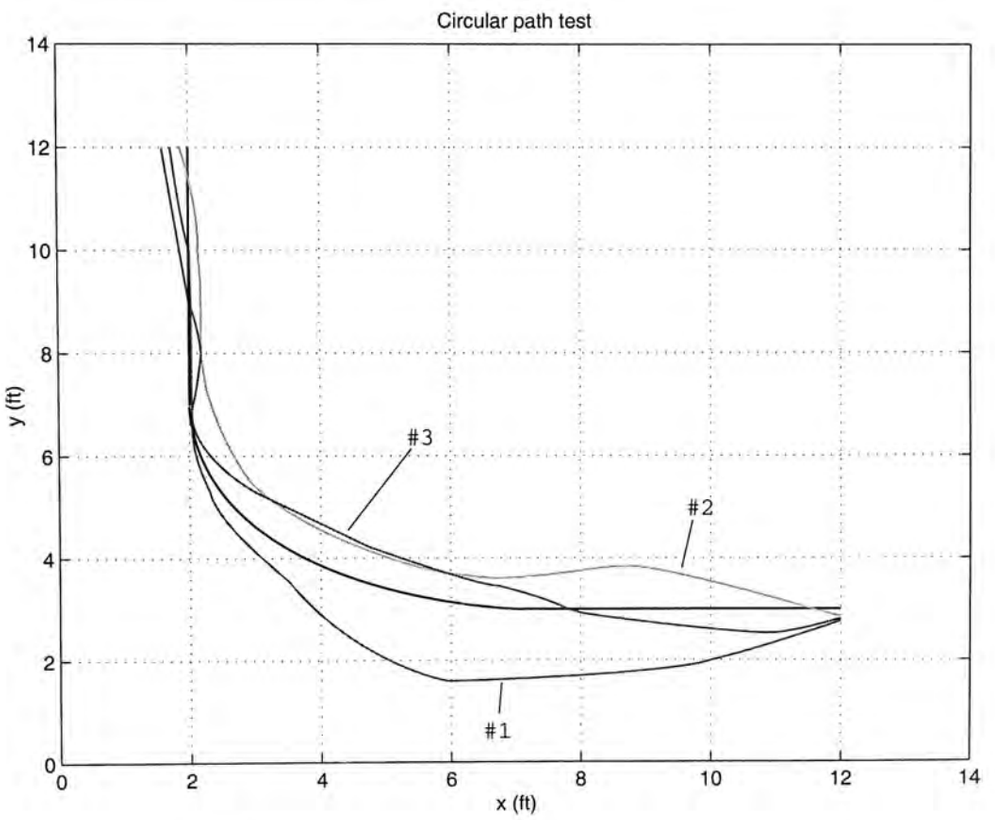


Figure 6.12: Gyrover trajectories in the curved path test.

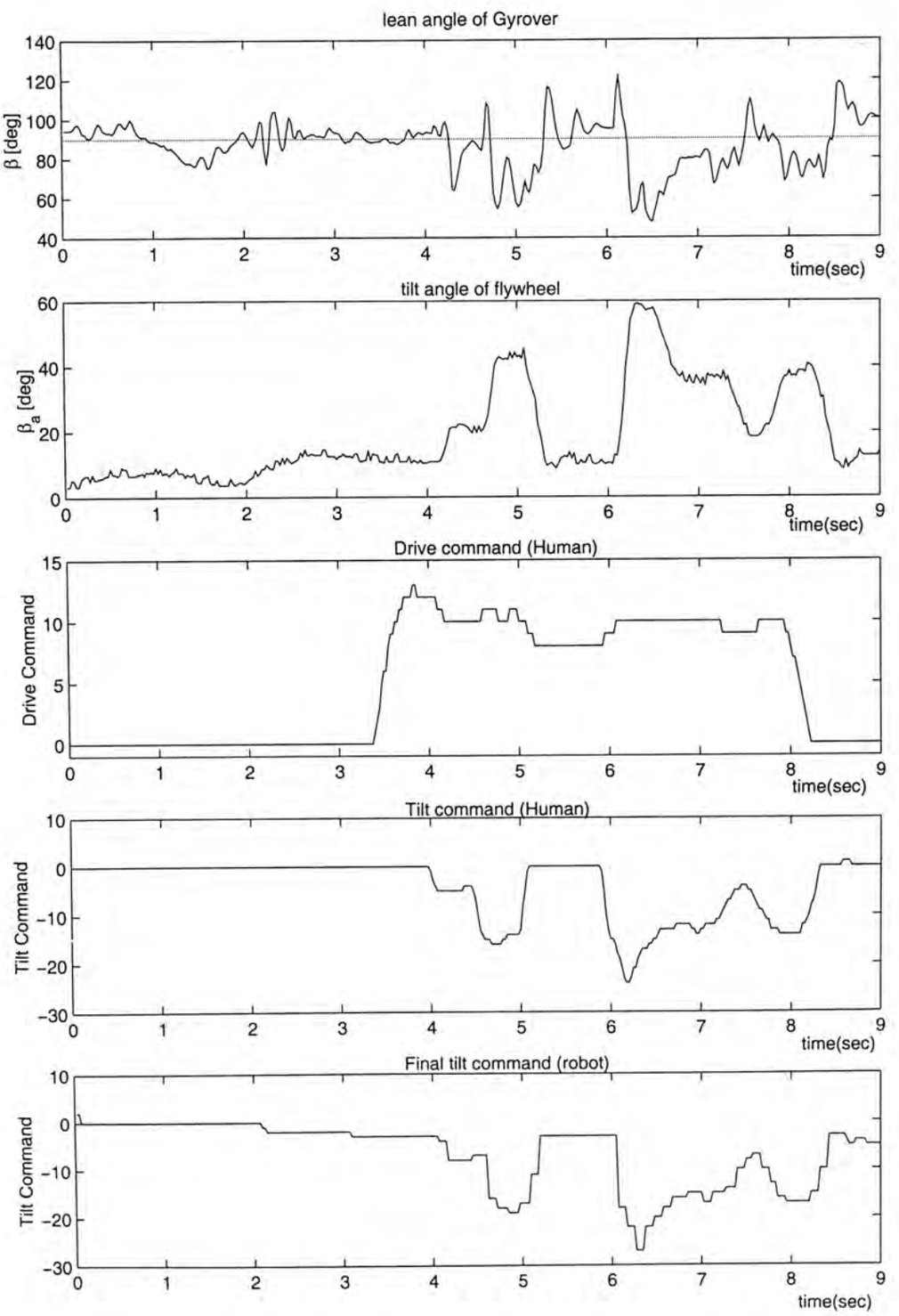


Figure 6.13: Sensor data acquired in the circular path test.

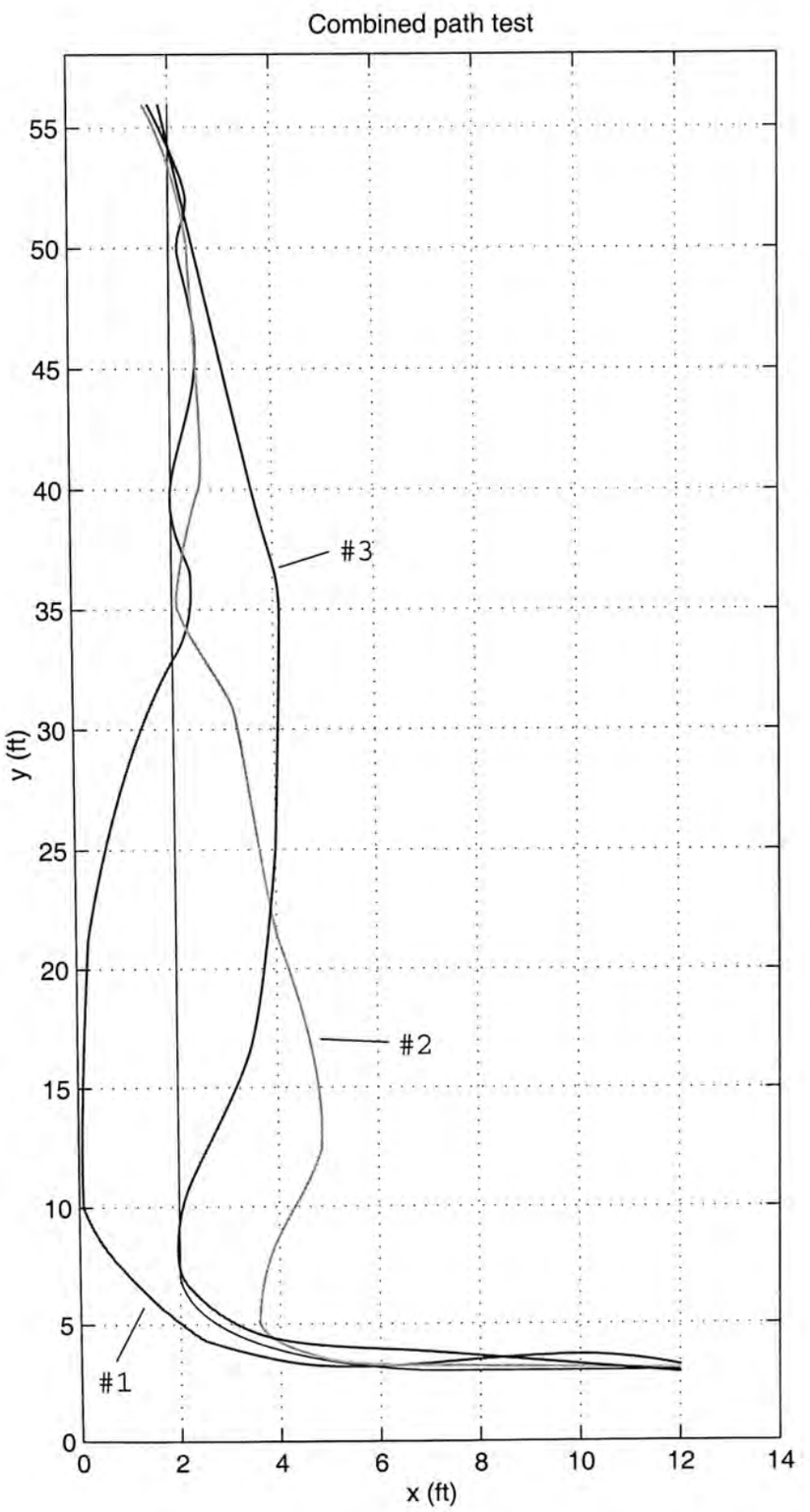


Figure 6.14: Gyrover trajectories in the combined path test.

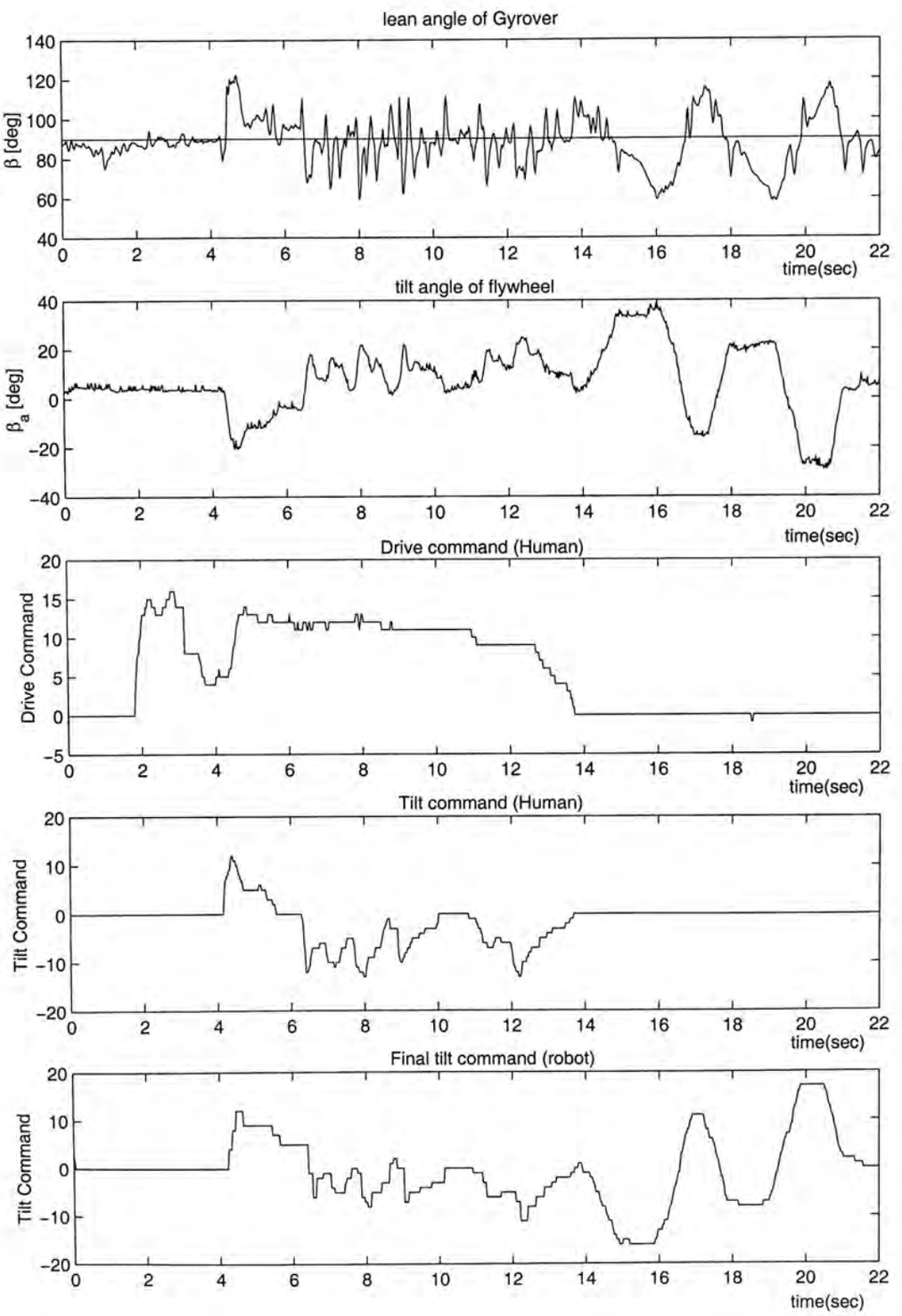


Figure 6.15: Sensor data acquired in the combined path test.

Chapter 7

Conclusion

7.1 Contributions

In this dissertation, we present a machine learning algorithm for Gyrover behaviors learning and a framework for Gyrover shared-control, which is original and unique from the previous work of the robot. We summarize the original contributions of this work below.

- We developed a behavior-based control architecture for Gyrover control. Under this architecture, the overall control task of Gyrover is decomposed into a number of behaviors. The subsumption architecture enables us to extend or to modify the existing system without affecting the original structure. Since the behaviors are distributed into differents levels of competence, the architecture enables the system to execute some high level goals while still servicing other low level behaviors. This control approach gives us a good foundation to develop a fully autonomous system for Gyrover in the near future.

- We propose an efficient neural-network learning architecture, cascade neural network with extended Kalman filtering, to model the human control behaviors in stabilizing and tiltup the robot into its upright positions. The instability

problem in the lateral direction of Gyrover, especially when the robot is being held in a static location, causes the main difficulty in Gyrover control for both human and machine. While some motions in Gyrover control are hard to obtain a complete mathematical model, this learning algorithm is an alternative method which is suitable to model the dynamic and complicated control strategy of human.

- We develop a shared control framework for Gyrover, based on the behavior-based architecture. Since building a fully autonomous system is costly and sometimes not practical, the main purpose of shared control is to reduce the operator's control burden in a complex system. In order to distribute the workload systematically in a shared control environment, we develop a decision function to let the system judges whether to execute the operator's command or not, by considering the Degree of Autonomy, the Strength of Conflict and the Confidence level. Experiments show that this shared control framework is able to share some of the control tasks from the operator without decreasing the maneuverability of the robot.

7.2 Future work

While this thesis provides a foundation for the development of Gyrover control system, it is certainly not the first and last word on this topic – it is only an important first step. There are a number of different directions in which the work in this thesis can be extended and applied. The followings are some possible improvements and extensions of this work.

First of all, although the lateral balancing of the robot can be obtained by

a machine learning algorithm, we still desire to seek for a better control method which is developed from the mathematical dynamic model. From the experiments, we observed that the existing lateral stablization model can only stable the robot around the vertical position with a tolerance of $\pm 10^o$, which may not be an ideal control if a high level of accuracy is required by the system.

From the behavior-based architecture of Gyrover, we have only implemented a small portion of control experimentally. In this thesis, we have successfully implemented the low level behaviors which enable the robot to retain its local stability. In the future, we are looking forward to have more implementations on different controls of Gyrover.

Furthermore, due to the lack of autonomous modules in the system, the system only allows a low level of sharing in the navigation tasks for the operator. In fact, the Gyrover control still relies heavily on the human operator. We suggest to equip the robot with one/two CCD camera(s) on board to work out some simple autonomous navigation control for the robot, which can assist the operator in controlling Gyrover more.

Finally, we are seeking the possibility of applying this shared control framework on other man-machine cooperating system, such as robotics wheelchair or semi-autonomous car driving. We believe that this framework can efficiently share the workload within the system for human and machine, while retaining a high level of maneuverability and flexibility of the orginal system.

Bibliography

[1] H. B. Brown and Y. Xu, "A Single-Wheel, Gyroscopically Stabilized Robot", *Proc. of the IEEE Int. Conf. on Robotics and Automation* , Vol. 4, pp. 3658-63, 1996.

[2] Gora C. Nandy and Y. Xu, "Dynamic Model of A Gyroscopic Wheel", *Proc. of the 1998 IEEE International Conference on Robotics and Automation* , Vol. 3, pp. 2683-88, 1998

[3] Shu-Jen Tsai, Enrique D. Ferreira, Christiaan J. J. Paredis, "Control of the Gyrover", *Proc. of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems* , pp. 179-184, 1999

[4] Y. Xu, K. W. Au, Nandy, G. C. and Brown, H. B., "Analysis of actuation and the dynamic balancing for a single wheel robot", *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems* , Vol. 3, pp. 1789-94, 1998.

[5] Kwok Wai Au, Yangsheng Xu, "Decoupled dynamics and stabiliztion of single wheel robot", *Proc. of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems* , Vol. 1, pp. 197-203, 1999.

[6] Kwok Wai Au, Yangsheng Xu, "Path following of a Single Wheel Robot", *Proc. of the 2000 IEEE International Conference on Robotics and Automation* , Vol. 3, pp. 2925-2930, 2000.

[7] Kwok Wai Au, Yangsheng Xu, "Dynamics and Control of a Single Wheel, Gyroscopically Stabilized Robot", M.Phil. Thesis, The Chinese University of Hong Kong, 1999.

[8] Yangsheng Xu, Loi Wah Sun, "Stabilization of a Gyroscopically Stabilized Robot on an Inlclined Plane", *Proc. of the 2000 IEEE International Conference on Robotics and Automation* , Vol. 4, pp. 3549-54, 2000.

[9] Yangsheng Xu, Loi Wah Sun, "Dynamics of a Rolling Disk and a Single Wheel Robot on an Inclined Plane", *Proc. of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems* , Vol. 1, pp. 811-816, 2000.

[10] R.A. Brooks, "A robust layered control system for a mobile robot", *IEEE Journal on Robotics and Automation*, Vol. RA-2 , No. 1, pp. 14-23, March 1986.

[11] M. Nechyba, Y. Xu, "Cascade Neural Networks with Node-Decoupled Extended Kalman Filtering", *Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, Vol. 1, pp. 214-9, 1997.

[12] M. Nechyba, Y. Xu, "Stochastic Similarity for Validating Human Control Strategy Models", *IEEE Transactions on Robotics and Automation*, Vol. 14 , No. 3, June 1998, pp. 437-451.

[13] M. Nechyba, "Learning and Validation of Human Control Strategy", Ph.D. Thesis, Carnegie Melon University, 1998.

[14] Y. Xu, W. Yu, K. Au, "Modeling Human Control Strategy in a Dynamically Stabilized Robot", *Proc. of the 1999 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Vol. 2, pp. 507-512, 1999.

[15] Montgomery, J. F., Bekey, G. A., "Learning helicopter control through "teaching by showing"", *Proc. of the 37th IEEE Conf. on Decision and Control*, Vol. 4, pp. 3647-52, 1998.

[16] S. Cherian, W.O. Troxel, M.M. Ali, "Design of a Behavior-based Micro-Rover Robot", *Proc. of the Intelligent Vehicles '92 Symposium*, pp. 280-287, 1992.

[17] R. Hartley, F. Pipitone, "Experiments with the subsumption architecture", *Proc. of 1991 IEEE International Conference on Robotics and Automation*, Vol.2, pp. 1652-58, 1991.

[18] M.A. Lewis, A.H. Fagg, G.A. Bekey, "The USC autonomous flying vehicle: an experiment in real-time behavior-based control", *Proc. of 1993 IEEE International Conference on Robotics and Automation*, Vol.2, pp. 422-429, 1993.

[19] T. Taipale, S. Hirai, "A behavior-based control system applied over multi-robot system", *Proc. of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems '93, IROS '93*, Vol.3, pp. 1941-43, 1993.

[20] E. Gat, A. Behar, R. Desai, R. Ivlev, J. Loch, D.P. Miller, "Behavior control for planetary exploration: interim report", *Proc. of the 1993 IEEE International Conference on Robotics and Automation*, Vol.2, pp. 567-571, 1993.

[21] J.F. Montgomery, A.H. Fagg, G.A. Bekey, "The USC AFV-I: a behavior-based entry in the 1994 International Aerial Robotics Competition", *IEEE Expert*, Vol.10, Issue.2, pp. 16-22, 1995.

[22] Y. Jeon, J. Park, I. Song, Y.J. Cho, S.R. Oh, "An object-oriented implementation of behavior-based control architecture ", *Proc. of the 1996 IEEE International Conference on Robotics and Automation*, Vol.1, pp. 706-711, 1996.

[23] N.O. Khessal, S.M.H. Amin, "Distributed behavior-based control architecture for a wall climbing robot ", *Proc. of the 1997 IEEE International Conference on Intelligent Engineering Systems, INES '97*, pp. 153-158, 1997.

[24] T. Balch, R.C. Arkin, "Behavior-based formation control for multirobot teams", *IEEE Transactions on Robotics and Automation*, Vol.14, Issue.6, pp. 926-939, 1998.

[25] M. Kasper, G. Fricke, E. von Puttkamer, "A behavior-based architecture for teaching more than reactive behaviors to mobile robots ", *The 1999 Third European Workshop on Advanced Mobile Robots, Eurobot '99*, pp. 203-210, 1999.

[26] L. Petersson, M. Egerstedt, H.I. Christensen, "A hybrid control architecture for mobile manipulation", *Proc. of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS '99*, Vol.3, pp. 1285-91, 1999.

[27] K. Watanable, K. Izumi, "A survey of robotic control systems constructed by using evolutionary computations ", *Proc. of the 1999 IEEE International Conference on Systems, Man, and Cybernetics*, Vol.2, pp. 758-763, 1999.

[28] T. Huntsberger, H. Aghazarian, E. Baumgartner, P.S. Schenker, "Behavior-based control systems for planetary autonomous robot outposts ", *Aerospace Conference Proceedings, 2000 IEEE*, Vol., pp. 679-686, 2000.

[29] R. Stenzel, "A behavior-based control architecture", *Proc. of on 2000 IEEE International Conference on Systems, Man, and Cybernetics*, Vol.5, pp. 3235-40, 2000.

[30] K.K. Lee, Y. Xu, "Human sensation modeling in virtual environments", *Proc. of 2000 IEEE/RSJ Internation Conference on Intelligent Robots and Systems*, Vol.1, pp. 151-156, 2000.

[31] D. A. Bell, S. P. Levine, Y. Koren, L. A. Jaros, J. Borenstein, "Design criteria for obstacle avoidance in a shared-control system", *Whitaker Student Scientific Paper Competition, RESNA '94 Annual Conference*, June, 1994.

[32] Tahboub, K.A., Asada, H.H., "A semi-autonomous control architecture applied to robotic wheelchairs", *Proc. of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1999. IROS '99.*, Vol.2, pp. 906-911, 1999.

[33] You Song; Wang Tianmiao; Wei Jun; Yang Fenglei; Zhang Qixian, "Share control in intelligent arm/hand teleoperated system", *Proc. of the 1999 IEEE International Conference on Robotics and Automation*, Vol.3, pp. 2489-94, 1999.

[34] Lee, S., Lee, H.S., "An advanced teleoperator control system. Design and evaluation", *Proc. of 1992 IEEE International Conference on Robotics and Automation*, Vol.1, pp. 859-864, 1992

[35] Chuanfan Guo, Tzyh-Jong Tarn, Ning Xi, Bejczy, A.K. , "Fusion of human and machine intelligence for telerobotic systems", *Proc. of the 1995 IEEE International Conference on Robotics and Automation*, Vol.3, pp. 3310-15, 1995.

[36] Aigner, P., McCarragher, B.J., "Modeling and constraining human interactions in shared control utilizing a discrete event framework", *IEEE Transactions on Systems, Man and Cybernetics, Part A*, Vol.30, Issue: 3, pp. 369-379, 2000.

[37] Cooper, R.A. , "Intelligent control of power wheelchairs", *IEEE Engineering in Medicine and Biology Magazine* , Vol.14, Issue : 4, pp. 423-431, 1995.

[38] Aigner, P., McCarragher, B., "Simultaneous human and autonomous control with constrained human action", *Proc. of the Australian and New Zealand Conference on Intelligent Information Systems, 1996.*, pp. 101-4, 1996.

[39] Yokokohji, Y., Ogawa, A., Hasunuma, H., Yoshikawa, T., "Operation modes for cooperating with autonomous functions in intelligent teleoperation systems", *Proc. of the 1993 IEEE International Conference on Robotics and Automation*, Vol.3, pp. 510-515, 1993.

[40] Aigner, P., McCarragher, B., "Shared control framework applied to a robotic aid for the blind", *Proc. of the 1998 IEEE International Conference on Robotics and Automation*, Vol.1, pp. 717-722, 1998.

[41] Simpson, R.C., Levine, S.P., "Adaptive shared control of a smart wheelchair operated by voice control", *Proc. of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'97*, Vol.2, pp. 622-626, 1997.

[42] Douglas, A., Y.S. Xu, "Real-time shared control system for space telerobotics", *Proc. of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems '93, IROS '93*, Vol.3, pp. 2117-22, 1993.

[43] G. Bourhis, Y. Agostini, "Man-machine cooperation for the control of an intelligent powered wheelchair", *Journal of Intelligent and Robotics Systems* , Vol.22, no.3-4 pp. 269-287, 1998.